#98

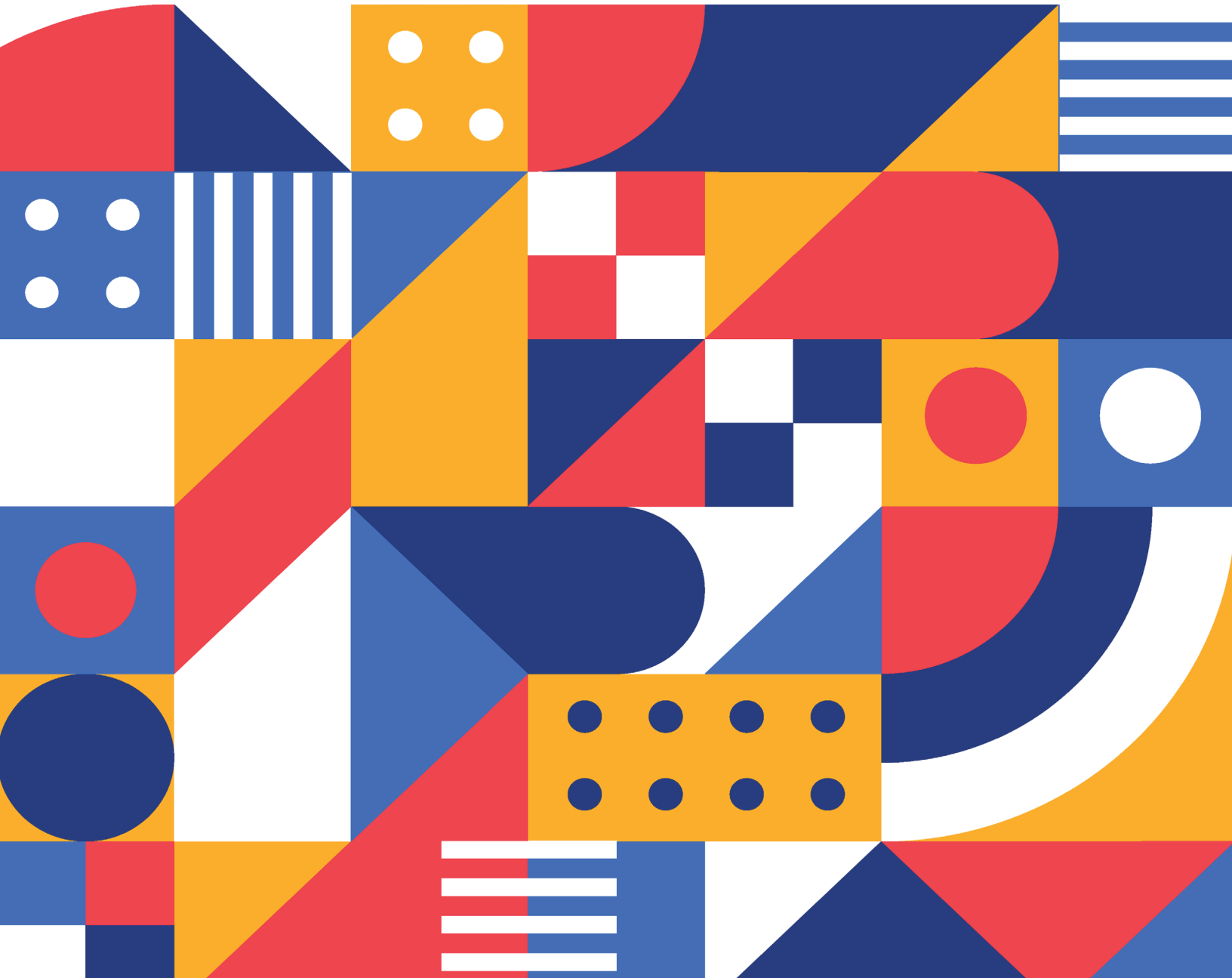
SERIE
MATEMÁTICAS

Sofía Natalia Galicia Haro

Notas para el curso inteligencia artificial

AÑO
2007

VÍNCULOS
MATEMÁTICOS



FACULTAD DE CIENCIAS

VÍNCULOS MATEMÁTICOS

Notas para el curso
INTELIGENCIA ARTIFICIAL

Sofía Natalia Galicia Haro*

VÍNCULOS MATEMÁTICOS NO. 98. 2012

(*) Departamento de Matemáticas, Facultad de Ciencias, UNAM
Impreso en la Coordinación de Servicios Editoriales de la Facultad de Ciencias, UNAM

NOTAS PARA EL CURSO

"INTELIGENCIA ARTIFICIAL"

Carrera Ciencias de la Computación

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Sofía Natalia Galicia Haro

Facultad de Ciencias
Departamento de Matemáticas
Ciudad Universitaria

2012

Estas notas fueron desarrolladas para el curso de introducción a la Inteligencia Artificial de la Facultad de Ciencias de la Universidad Autónoma de México, de la carrera de Ciencias de la Computación.

Estas notas de curso pretenden dar una visión general de los principales temas de IA: búsqueda, razonamiento, aprendizaje. El objetivo de este material, es servir como apoyo para la enseñanza de un curso introductorio a la IA a nivel licenciatura y están basadas principalmente en las referencias bibliográficas indicadas al final de estas notas.

CONTENIDO

1. Introducción

1.1. ¿Qué es la Inteligencia Artificial?

Hemos oído mucho esta combinación de palabras en:

- El cine
- La televisión
- Notas periodísticas
- etc.

Nos puede confundir el sentido que se implica en cada caso. Para los que estudiamos Ciencias de la Computación, cuando hablamos de Inteligencia Artificial, en general, lo que intentamos es resaltar el sentido de inteligencia creada por el ser humano y colocada en máquinas de su creación. Entonces, podríamos considerar que es la rama de la computación que trata de lograr que las computadoras hagan cosas que cuando las hacen los seres humanos requieren inteligencia.

¿Porqué estudiar este tema?

Existimos los humanos y creemos que somos inteligentes, entonces parece factible establecer cómo funciona la inteligencia y el razonamiento humano si tenemos un modelo. Una vez que sepamos cómo funciona totalmente la inteligencia en el ser humano será posible implantarla en una computadora de manera similar. En el área de Inteligencia Artificial que aplica métodos bio-inspirados se considera que aún sin que se conozca al detalle el funcionamiento de la inteligencia se podrá lograr implantarla. Para esto se requiere la participación de diversas disciplinas que aporten métodos para describir el funcionamiento del cerebro humano, es un campo donde hay mucho por hacer. Como es una tarea demasiado compleja se ha dividido en distintas áreas, por ejemplo: Lenguaje natural, Sistemas expertos, Robótica, Máquinas con Razonamiento, Visión artificial, Aprendizaje automático, etc.

¿Cómo abordar este reto?

Si la inteligencia es lo que se resalta y hace la diferencia con otros sistemas computacionales, primero deberíamos caracterizar la inteligencia, conocer sus características principales, cómo definirlas, cómo representarlas. Esto resulta muy complejo y ha sido motivo de estudio desde hace siglos.

Temas de introducción a la Inteligencia Artificial

Este curso es introductorio al área de Inteligencia Artificial, cada uno de los temas considerados es muy amplio. Para ahondar en cada tema pueden referirse a la bibliografía al final de estas notas y en esas referencias encontrarán a su vez más información sobre temas específicos.

De los temas clásicos para una introducción al área de Inteligencia Artificial se abordan los siguientes:

- Búsqueda, para resolver problemas
- Representación del conocimiento
- Aprendizaje
- Temas de Lenguaje natural

EJERCICIO: ¿Qué sistema inteligente conozco? ¿Por qué es inteligente?

1.2. ¿Qué es la inteligencia?

Entre otras definiciones [Ginsberg, 1984] indica que la inteligencia artificial es la comprensión y construcción de entidades inteligentes. Así que debemos saber qué es la inteligencia y qué es una entidad inteligente para intentar reproducirlas. Si los humanos somos los más inteligentes, debemos entender qué es la inteligencia en un humano y cómo se puede implementar en una máquina. Pero aún no conocemos en detalle la inteligencia humana.

Desde hace más de 2000 años, los filósofos han intentado comprender cómo aprendemos los seres humanos, cómo razonamos, cómo recordamos. Platón ya se preguntaba como diferenciar entre piedad y no piedad (Eutifrón de los Primeros diálogos), cómo definirla. Aristóteles, su alumno, desarrolló reglas para establecer un razonamiento, el sistema formal de silogismos, pero también consideraba la intuición.

Descartes establece la diferencia entre mente y materia y los problemas que origina (Teoría de las dos sustancias). Si la mente está completamente sujeta a las leyes físicas entonces no hay libre albedrío. Pero en el hombre hay *dualismo* una parte de la mente (espíritu, alma) que está al margen de la naturaleza. Una clase era la sustancia pensante, o inteligencia, y la otra la sustancia extensa, o física. Descartes dio el método para conducir el razonamiento y las reglas utilizables con el objeto de evitar el error (Discurso del Método para dirigir bien la Razón y buscar bien la verdad en las ciencias).

John Stuart Mill promovió la idea del criterio de decisión racional en todos los ámbitos de la actividad humana. Por ejemplo, se prueba que el arte médico es bueno por su conducción a la salud; ¿pero cómo es posible demostrar que la salud es algo bueno? El arte musical es bueno, por los motivos, entre otros, de que produce placer; ¿pero qué prueba es posible dar para establecer que el placer es bueno? Para dar una repuesta que por supuesto no dependa de un impulso ciego o una elección arbitraria, el tema se sitúa dentro de la cognizance de la facultad racional.

Para pasar de la filosofía a una ciencia formal los matemáticos han aportado métodos como la lógica y la probabilidad. En lógica: Hilbert, en su trabajo de 1899 sobre los fundamentos de geometría fue el primero en describir un método sistemático para demostrar la no-deducibilidad en la lógica. En lógica matemática, los dos célebres teoremas de incompletitud de Gödel se refieren a la existencia de aseveraciones verdaderas indecidibles (no es posible definir su validez mediante ningún algoritmo) y a que ningún sistema consistente se puede usar para demostrarse a sí mismo, lo que demuestra que la aritmética básica no se puede usar para demostrar su propia consistencia, y por lo tanto tampoco puede demostrar la consistencia de nada más fuerte.

La psicología cognoscitiva cuya característica fundamental es que el cerebro posee y procesa información. William James (1842-1910), uno de los representantes más importantes del pragmatismo. En el conductismo, K. J. W. Craik (1943), encontró 3 condiciones fundamentales para los agentes basados en conocimiento: 1) el estímulo traducido a una representación interna, 2) la representación se manipula mediante procesos cognoscitivos, 3) éstas se traducen en acciones.

1.3. ¿Pueden pensar las máquinas?

¿Cómo determinar si alguien es inteligente? Si hacemos una pregunta a un grupo de personas, podríamos considerar que quién dé más rápido la respuesta correcta es inteligente. Si hacemos varias preguntas, podríamos considerar que lo será quién dé más cantidad de respuestas correctas. En IA se considera que el comportamiento inteligente involucra percibir, razonar, aprender, comunicarse, actuar en entornos complejos.

¿Cómo determinar si una máquina es inteligente? Alan M. Turing (Turing, 1950), expone sus ideas para responder a la pregunta: ¿Pueden pensar las máquinas? En su artículo, propone una prueba que tal vez no de una manera correcta[1], se ha considerado como una prueba para demostrar la existencia de inteligencia en una computadora. Para responder a esa pregunta, Turing propone el "juego de la imitación", este juego tiene los siguientes elementos:

3 participantes:	hombre (A), mujer (B), interrogador (C)
Comunicación:	teletipo
Objetivo:	C debe determinar cuál es el hombre (Y) y cuál es la mujer(X); X es A, Y es B o al revés
Operación:	Al Interrogador se le permite hacer preguntas a A y B (ejem: ¿puede decirme la longitud de su cabello?
Objetivo de A:	intentar que C se equivoque
Objetivo de B:	ayudar a C

¿Y si una máquina interpreta el rol de A? Turing sugiere que si las respuestas de la computadora fueran indistinguibles de las de un humano, podríamos decir que la computadora piensa.

Turing, en su artículo, predijo que las máquinas podrían eventualmente pasar la prueba. De hecho, estimó que por el año 2000, las máquinas con cerca de 119Mb de memoria podrían engañar al 30% de jueces humanos durante una prueba de cinco minutos. También predijo que para entonces la gente no consideraría contradictoria la frase "máquina pensante.

El Concurso LOEBNER[2] ha prometido un premio de $100,000 dólares y una medalla de oro para la primera computadora cuyas respuestas sean indistinguibles de las de un ser humano. Cada año da un premio anual de $2000 y una medalla de bronce al sistema que se parezca más en sus respuestas a un ser humano. Joseph Weizenbaum, en1966 (en ACM) describe su sistema ELIZA, siguiendo esta idea de Turing.

1.4. Parámetros para definir sistemas de IA

Actualmente, los esfuerzos de la IA están encaminados tanto a la construcción de entidades inteligentes como a su comprensión. Para analizar la complejidad de los sistemas de IA, [Russell y Norvig, 2004] proponen los siguientes parámetros que se han considerado en distintas definiciones de lo que es la IA:

Como humano	Racionalmente
PENSAR	PENSAR
ACTUAR	ACTUAR

Los autores consideran los parámetros: Actuar y Pensar. Estos parámetros se analizan considerando si se trata de sistemas inteligentes que imitan al ser humano o sistemas inteligentes que pretenden actuar y pensar de una forma racional. Esta clasificación permite separar los sistemas de inteligencia artificial en dos grupos: uno muy complejo donde se intenta implementar el pensamiento y la actuación de los seres humanos en general, y otro grupo de sistemas inteligentes limitados a la parte racional que tenemos los seres humanos. Aún con esta separación, sigue siendo difícil la implementación de agentes, es decir, de sistemas inteligentes racionales.

---

[1] http://plato.stanford.edu/entries/turing-test/

[2] http://www.loebner.net/Prizef/loebner-prize.html

Pensar, se refiere a procesos mentales y de razonamiento, actuar se refiere a la conducta asumida.

Como humano

Actuar: Prueba de Turing

Una máquina que actuara como ser humano tendría que engañar a un evaluador humano, haciéndole creer que se trata de un humano y no de una máquina. Esto es muy complejo, por ejemplo ¿creeríamos que un ser humano puede realizar operaciones aritméticas de cifras estratosféricas en dos o tres segundos?, ¿que nunca se equivoca?, etc. Turing consideraba un interrogatorio por teletipo pero en el siglo XXI quisiéramos que físicamente fuera similar a un ser humano, con capacidades de visión, voz, razonamiento, aprendizaje, etc.

Pensar: ¿Cómo se desarrolla dentro de la mente humana?

Para saber cómo pensamos los seres humanos se requieren teorías científicas sobre todas las actividades internas del cerebro. Para validarlas se deberían predecir y comprobar las funciones cerebrales. La introspección y los experimentos psicológicos se han empleado para intentar describir el funcionamiento humano.

Entre las ciencias que podrán desarrollar esas teoría, la sicología cognitiva intenta proporcionar una explicación científica de cómo el cerebro lleva a cabo sus funciones mentales como la visión, la memoria, el lenguaje y el pensamiento; la ciencia cognitiva estudia cómo la información se representa y transforma en la mente; y la neurociencia cognitiva que estudia los mecanismos biológicos subyacentes a la cognición. El término cognición se ha usado de modos diferentes. En la sicología se refiere a la forma del proceso de información de las funciones psicológicas de un individuo. Otras interpretaciones lo unen al desarrollo de conceptos, al entendimiento y tentativa de entender el mundo.

Racionalmente

Pensar: la manera correcta de pensar.

Aristóteles, famoso por sus silogismos (Sócrates es un hombre, todos los hombres son mortales, por lo tanto Sócrates es mortal), fue uno de los primeros en intentar delinear la teoría del pensamiento correcto que distingue los falsos modos de razonar. Con la lógica se suministra una notación precisa y reglas para representar cualquier objeto del mundo y para razonar con ellos.

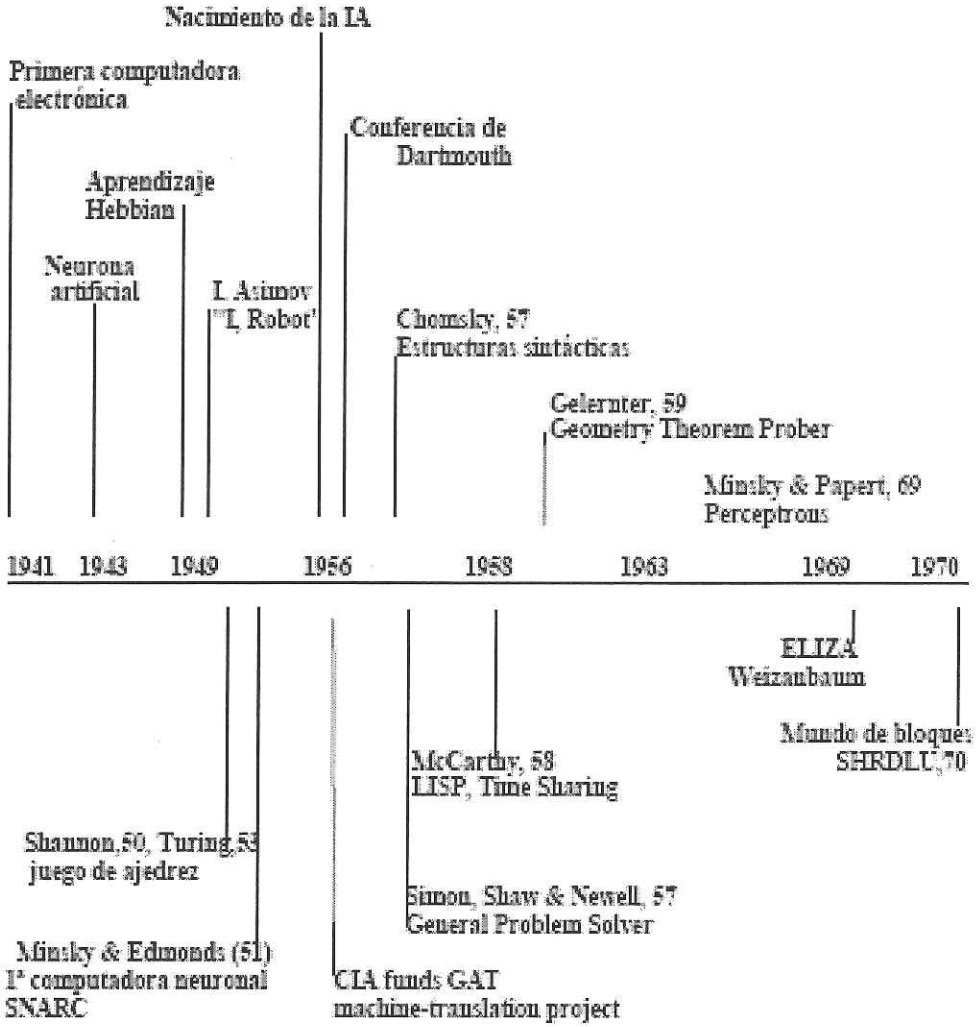
Sin embargo, la lógica no ha solucionado el problema de representar cualquier conocimiento informal, especialmente cuando el conocimiento tiene incertidumbre. Otro problema es razonar en la práctica con cientos de miles de oraciones.

Actuar: comportamiento correcto

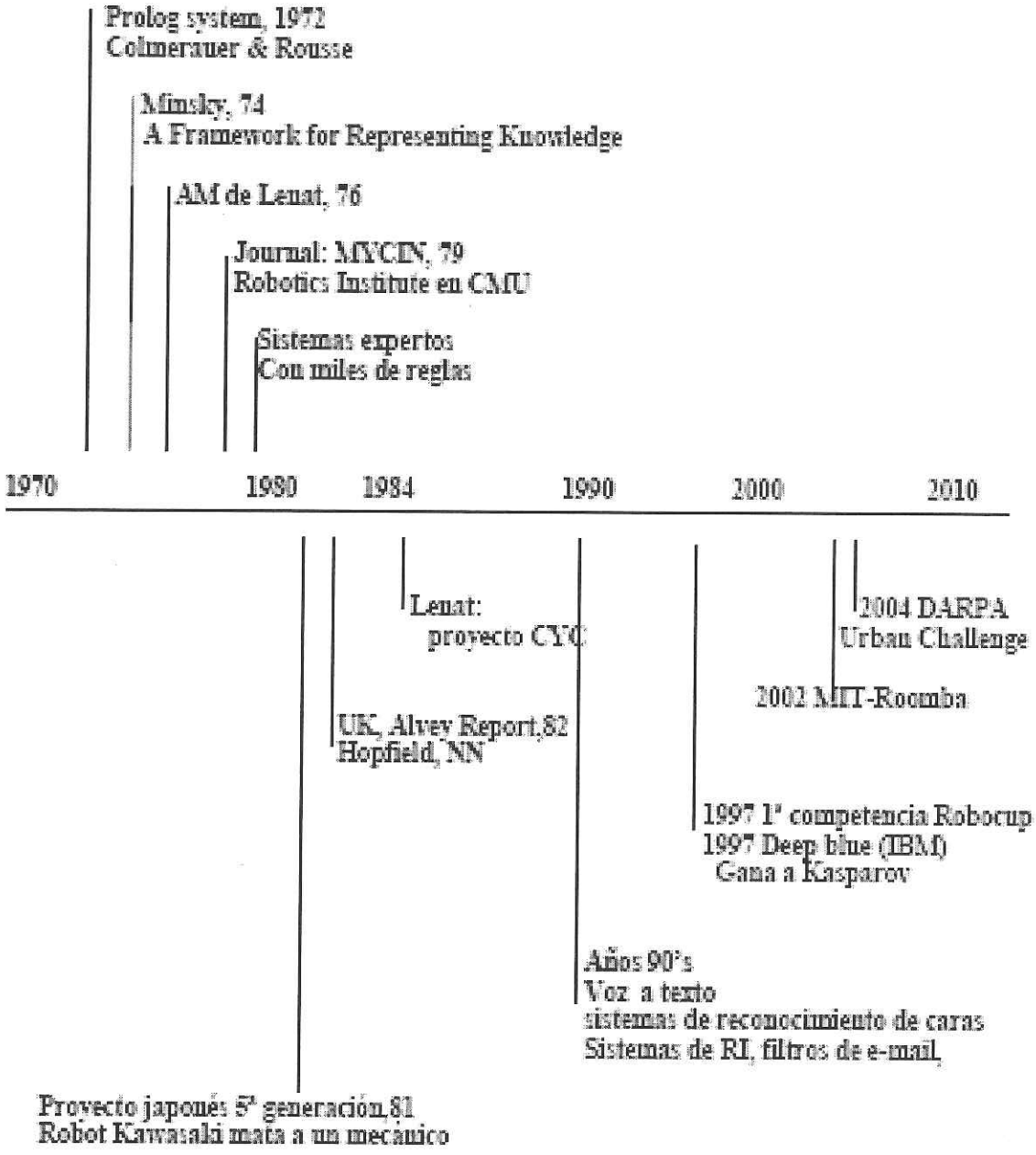
Hacer siempre lo correcto no es posible en entornos complejos, por ejemplo: no lastimar a la gente (Yo Robot, de Isaac Asimov). Una forma correcta sería lograr los objetivos deseados basándose en ciertos supuestos, esto llevaría a la toma de decisiones perfectas. Sin embargo, las acciones correctas no siempre depende de racionalidad, por ejemplo: la mamá que tiene frío y le pone suéter a su hijo. A veces los actos reflejos son mejores que una acción basada en una deliberación, por ejemplo: retirar una mano que se acercó a la hornilla prendida de una estufa.

EJERCICIO: Seleccionar un aparato común y pensar en cómo se modificaría para que tuviera inteligencia artificial. Definir sus tareas de acuerdo con los parámetros indicados.

1.5. Historia – Siglo XX



1917: Karel Capek acuñó el término *robot*, en checo 'robot' significa 'trabajador'.
1928: John von Neumann presentó su teorema minimax, empleado en programas de juego de adversarios
1952 Arthur Samuel creó un programa para jugar damas, en 1955 le añadió características que le permitieran aprender, aprendió a nivel de torneo.
1963 DARPA apoyó financiando el proyecto MAC en MIT, con dos millón de dólares. El proyecto MAC fue famoso por la investigación innovadora en sistemas operativos, inteligencia artificial, y la teoría de la computación.



¿Inteligencia en robots?

- http://world.honda.com/ASIMO/technology/
- Creepy Robot Baby CB2

¿Inteligencia en juegos?

- Video juegos: Battlecruiser 3000AD, Creatures (artificial life program)
- Ajedrez: Deep Blue, Deep Fritz

2. AGENTES

2.1. ¿Qué es un agente?

Un agente es una estructura que percibe su entorno a través de sensores y actúa en ese entorno a través de efectores. La función del agente mapea de secuencias de percepciones a acciones. Para producir esa función del agente, el programa del agente se desarrolla en una arquitectura física que hace que las percepciones obtenidas por los sensores estén disponibles para el programa, corra el programa y alimente las acciones seleccionadas por el programa a los efectores conforme se generen.

Un agente robot que carga objetos de un cuarto a otro, usaría cámaras y detectores de sonido como sensores, usaría dos brazos mecánicos y un motor para deslizarse, como efectores. Su programa incluiría las reglas para moverse, y para mover la carga. Un agente de software codifica cadenas de bits como percepciones y acciones, su programa es el software que implementa las reglas de operación.

Un agente humano usa como sensores: ojos, oídos, etc. y como efectores: manos, piernas, etc.

Hay muchos tipos de agentes y diferentes entornos a considerar. Por ejemplo:

- Agentes para condiciones de menor gravedad.
- Agentes para hacer limpieza de cuartos
- Agentes para trabajar en Internet
- etc.

Aún en dominios limitados del "mundo real" serían agentes complejos por lo que en muchos ejemplos realmente se consideran "agentes de juguete" en un mundo espacial cuadriculado.

2.2. Agente racional

Un agente racional es un agente que hace lo correcto. Sus acciones se basan en lo que percibe y las acciones que puede realizar. Una acción correcta es la que logra que el agente tenga éxito.

La definición de un agente racional ideal: para cada secuencia posible de percepciones, un agente racional ideal debería hacer cualquier acción esperada para maximizar su medida de rendimiento, basándose en la evidencia provista por la secuencia de percepciones y por todo el conocimiento incorporado que tiene el agente.

Esta definición no quiere decir que está bien cruzar una calle sin voltear a ambos lados de la calle para llegar más pronto al otro extremo, esto implicaría un riesgo. Un agente racional debería tener la acción "mirar a ambos lados" en un paso anterior a cruzar la calle, ya que esa acción maximiza el funcionamiento esperado, llegar al otro extremo sano y salvo.

Racionalidad y sucesos esperados

Un agente omnisciente[3] conoce el resultado actual de sus acciones y puede actuar en correspondencia, pero en la realidad esto no es posible. Salimos de nuestra casa y vemos libre la banqueta, entonces caminamos sobre la acera y de repente se abre una puerta automática que opera con gran velocidad: nos golpea, o alguien tira basura por la ventana: nos ensucia. Esta acción no es

[3] Omnisciencia, que tiene sabiduría o conocimiento de muchas cosas. En este caso se considera que sabe todo con conocimiento infinito

irracional, nos indica que la racionalidad tiene que ver con sucesos esperados dadas las percepciones obtenidas. Además puede ser que realmente no podamos detener la puerta ni la basura.

Lo que es racional en un tiempo dado depende de lo siguiente:

- la medida de desempeño que define el grado de éxito
- todo lo que el agente percibe (historia completa de percepciones)
- lo que conoce previamente el agente del entorno (o ambiente)
- las acciones que el agente puede realizar

El agente racional podrá ser autónomo si su comportamiento se determina por su propia experiencia, con la habilidad para aprender y adaptarse al ambiente. El agente aprenderá del ambiente, por ejemplo que hay puertas que abren con gran velocidad hacia la banqueta. Los agentes con éxito dividen la tarea de calcular la función del agente en tres períodos: cuando el agente se diseña, cuando delibera su siguiente acción y cuando aprende. Si solamente se basa el agente en el conocimiento suministrado durante su diseño, no tendrá autonomía.

Medidas de desempeño

¿Cómo definir el éxito del agente? Es necesario evaluar objetivamente sus logros, basándose en lo que se espera en el ambiente dado. Por ejemplo: un agente que lava ropa. Si queremos como meta que la ropa esté libre de suciedad. ¿Qué pasaría con la ropa que tiene manchas difíciles de eliminar? El agente podría pasarse lavando la ropa sin parar.

La medida de desempeño debe considerar: cantidad de ropa lavada, cantidad de jabón utilizado, cantidad de agua utilizada, cantidad de tiempo empleado, mantenimiento de la zona de lavado, etc.

La selección de la medida de desempeño no es fácil y en muchos casos los elementos considerados son contradictorios.

EJERCICIO – Analizar la medida de desempeño de un agente que atiende clientes en un servicio de telefonía celular.

Secuencias de percepciones para decidir acciones

Si el comportamiento de un agente depende solamente de la secuencia de sus percepciones entonces se puede describir el programa del agente haciendo una tabla de las acciones que toma en respuesta a cada posible secuencia de percepciones. El programa del agente recibe sólo una percepción como entrada. Es decisión del agente construir la secuencia de percepciones en memoria.

En tareas limitadas, es posible encontrar una función que relacione las percepciones con las acciones y el programa del agente se vuelve compacto. Sin embargo, la lista de secuencias de percepciones puede ser muy larga, incluso infinita en algunos casos. Por ejemplo: las secuencias de percepciones posibles de un juego de ajedrez entre grandes maestros es muy grande.

Las desventajas de considerar una tabla de secuencias de percepciones son: 1) la tabla puede requerir un tamaño enorme, 2) escribir la tabla completa tomaría mucho tiempo, si es enorme, 3) el agente no tendría autonomía, si el entorno cambia el agente estaría perdido, 4) si por el contrario sólo tuviera un mecanismo de aprendizaje, podría necesitar una eternidad para aprender el valor correcto de cada entrada en la tabla, dependiendo del mecanismo de aprendizaje.

Si las acciones de los agentes se basan completamente en conocimiento incorporado, de tal forma que no se tomen en cuenta sus percepciones, el agente carece de autonomía. El comportamiento de un agente puede basarse tanto en su experiencia como en el conocimiento incorporado usado en la construcción del agente para el entorno particular en el cual opera. Un sistema es autónomo en la medida en que su comportamiento se determina por su propia experiencia. Sería demasiado estricto

requerir autonomía completa ya que cuando el agente tiene poca o ninguna experiencia, tendría que actuar aleatoriamente a menos que el diseñador le diera alguna asistencia.

Un agente inteligente verdaderamente autónomo debería ser capaz de operar con éxito en una amplia variedad de entornos, dado el tiempo suficiente para adaptarse.

2.3. Especificación del agente

Para diseñar un agente racional, que podemos calificar como inteligente, se deben tomar en cuenta precisamente los elementos para ser racional que le permitirán realizar su tarea en el ambiente considerado.

- Tipo de agente
- Meta (medida de desempeño)
- Percepciones
- El entorno
- Acciones que realizará

Por ejemplo:

Tipo de agente:	robot que limpia cuartos	vendedor de boletos de tren
Percepciones:	pixels de imágenes	palabras escritas
Acciones.	levantar basura	desplegar itinerarios
	Barrer	hacer preguntas
Metas:	cuarto libre de basura	usuario compra boletos
Entorno:	dentro de edificio	Máquina interactiva
		Terminales de tren

EJERCICIO Describir un agente como el carro KITT de la televisión

2.4. Entorno

El tipo de ambientes en que los agentes racionales pueden operar es muy amplio y su complejidad determinará el diseño del agente. Para caracterizar el entorno [Norvig & Russell] han considerado las siguientes dimensiones del entorno:

Accesible vs. no accesible

Si el aparato sensorial del agente le da acceso al estado completo del entorno, se dice que el entorno es accesible al agente. Un entorno es efectivamente accesible si los sensores detectan todos los aspectos que son relevantes para la selección de la acción.

Determinístico vs. no determinístico

Si el siguiente estado del entorno está completamente determinado por el estado actual y las acciones seleccionadas por el agente, se dice que el entorno es determinístico. Si el entorno es accesible y determinístico el agente no se tiene que preocupar de la incertidumbre.

Si el entorno es inaccesible, puede parecer no determinístico. Así que a menudo es mejor pensar en el entorno como determinístico o no determinístico desde el punto de vista del agente.

Episódico vs. no episódico

En un entorno episódico, el comportamiento del agente se divide en episodios. Cada episodio consiste del agente percibiendo y luego actuando. La calidad de su acción depende del episodio mismo, porque episodios subsecuentes no dependen de las acciones que ocurren en episodios previos. Entornos episódicos son más simples porque el agente no necesita pensar por adelantado.

Estático vs. dinámico

Si el entorno puede cambiar cuando el agente está deliberando, se dice que es un entorno dinámico para el agente, si no, es estático. Los entornos estáticos son fáciles de tratar porque el agente no tiene que observar lo que sucede en el mundo al mismo tiempo que decide una acción, ni necesita preocuparse por el paso del tiempo.

Discreto vs. continuo

Si hay un número limitado de distintas pero claramente definidas percepciones y acciones, se dice que el entorno es discreto. El juego de ajedrez es discreto (número fijo de posibles movimientos), KITT es continuo, la velocidad y la posición del carro se extienden a través de un rango de valores continuos.

Diferentes tipos de entornos requieren programas de agente diferentes para tratarlos eficazmente. El caso más difícil: inaccesible, no episódico, dinámico y continuo. La mayoría de las situaciones reales son tan complejas que aunque sean deterministas para propósitos prácticos, deben tratarse como no-determinísticos. A continuación algunos ejemplos de Entornos:

Agente \ Entorno	Accesible	Determinístico	Episódico	Estático	discreto
Kitt	No	no	No	no	no
Mundo de bloques	Si	si	Si	si	si
Poker	No	no	No	si	si

Las respuestas pueden variar dependiendo de cómo se conceptualicen, por ejemplo KITT puede considerarse discreto porque la imagen de la cámara está digitalizada lo cual lleva a valores de pixels discretos. Si el mundo de bloques tiene un espacio abierto y otras personas pueden intervenir el entorno será no determinístico.

EJERCICIO: Seleccionar una tarea que pudiera realizar un robot y que nosotros conocemos bien, caracterizar el entorno.

2.5. Estructura de agentes

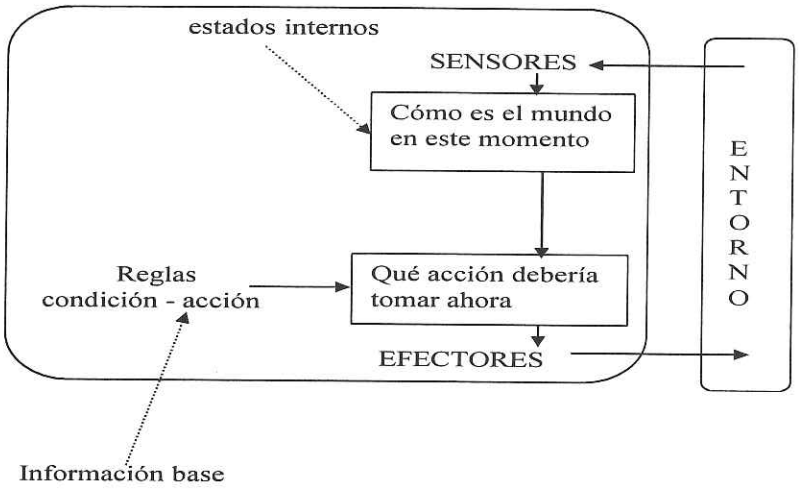
A continuación se describen cuatro tipos básicos de estructuras de agentes en orden ascendente de mayor generalidad.

- Agente simple reflejo
- Agente reflejo basado en modelos
- Agente basado en metas
- Agente basado en utilidad

Agente reflejo

El proceso del agente es un ciclo percepción-acción, es decir, puramente reactivo. Las decisiones del agente se basan únicamente en sus percepciones y el estado actual. Estos agentes se pueden implantar muy eficientemente pero su rango de aplicabilidad es muy limitado. Si el agente no tiene un sensor que le permita ver si le cae algo desde una altura mayor a la suya, no se librará de posible basura cayéndole encima.

Los humanos tenemos muchas reacciones de este tipo, algunas aprendidas otras innatas. Si acercamos una mano al fuego, la percepción es dolor y la acción será retirar la mano del fuego.



Agente reflejo

Agente reflejo basado en modelos

Agentes más complejos que recuerdan propiedades y almacenan modelos del mundo. Modelo se emplea en el sentido más amplio, como cualquier estructura simbólica que se correlacione suficientemente con el mundo real. Por ejemplo: cálculo de la distancia recorrida dada la velocidad.

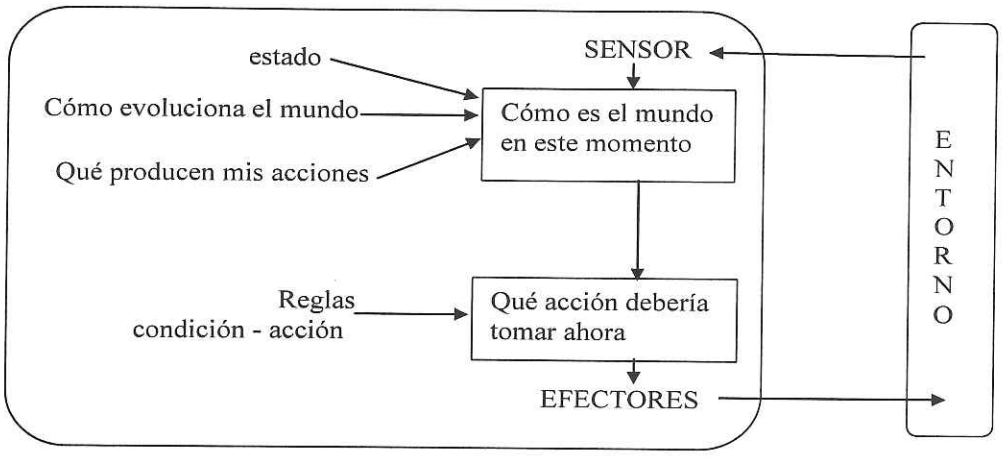
Pueden manejar la falta de accesibilidad del entorno, manteniendo información de la parte del mundo que no pueden ver. Estos agentes razonan mediante deducciones de propiedades del mundo.

Para actualizar la información del mundo se requiere información: 1) de cómo evoluciona el mundo de forma independiente al agente, por ejemplo una puerta abriendo hacia la banqueta estará más cerca del agente que un momento anterior, 2) de cómo las acciones del agente afectan al mundo, por ejemplo el agente estará (a un paso de 6 km/h) un kilómetro adelante después de diez minutos.

Volviendo al ejemplo de un agente que va caminando por la calle cuando rápidamente se abre una puerta automática de una casa: el agente necesita llevar un registro de la posición de las puertas si no las puede ver ya en relación a su posición.

Agentes basados en metas

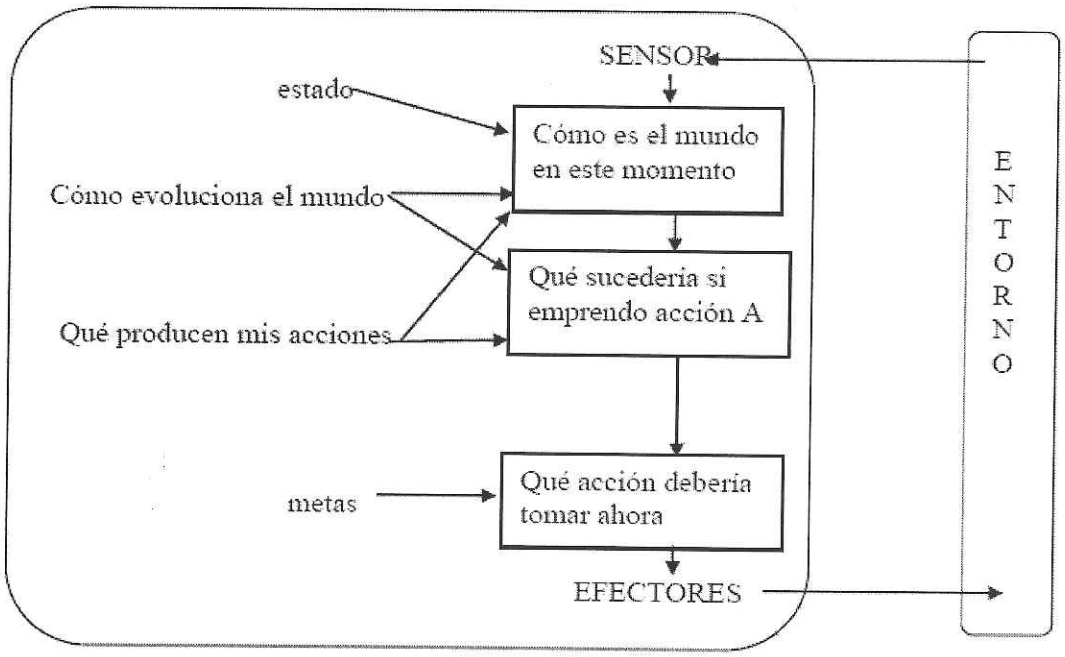
Conocer el estado actual del entorno no es suficiente para decidir qué hacer. Por ejemplo: la decisión correcta del agente que va por la calle en un cruce determinado depende de adónde quiere llegar. Si se conoce la información de la meta, ésta contribuirá a la decisión de las acciones, combinada con los resultados de posibles acciones.



Agente reflejo con estado interno

Si la meta resulta de una sola acción, el camino es directo pero otras veces puede ser tortuoso, considerando largas secuencias de vueltas y giros para encontrar una forma de lograr la meta. Búsqueda y planeación son las subáreas de la IA para encontrar secuencias que logren las metas del agente.

Aunque los agentes basados en metas parecen menos eficientes, son más flexibles. Por ejemplo, para alcanzar diferentes destinos, las reglas del agente reactivo funcionan para un destino, las reglas deben reemplazarse para ir a un punto nuevo. El agente puede ir de norte a sur pero puede decidir ir al oeste si el camino directo está obstruido.



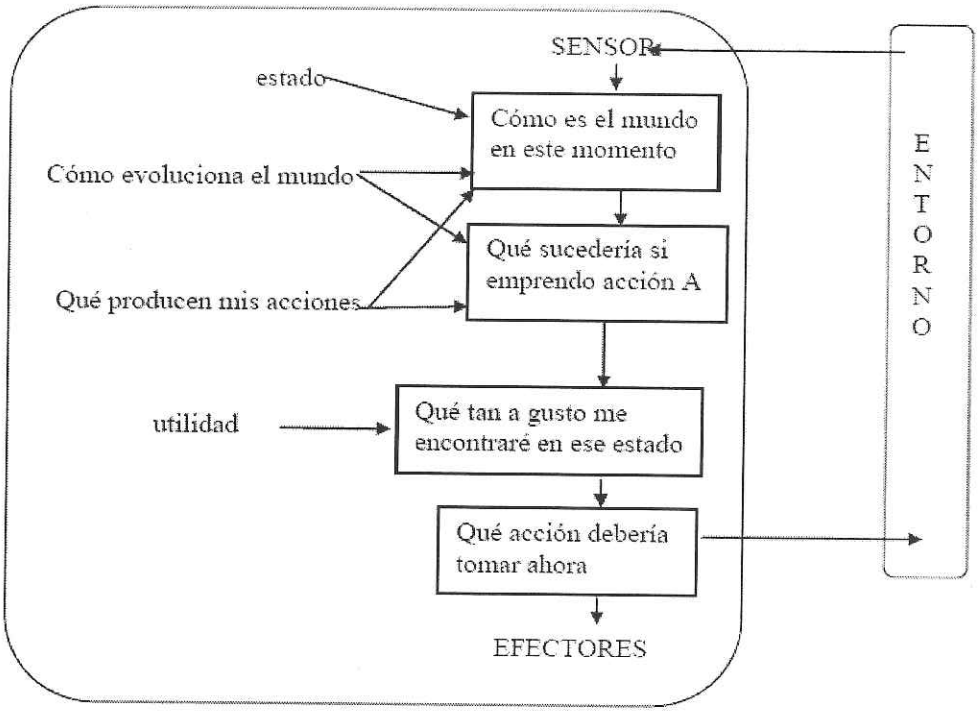
Agente basado en metas

Agentes basados en la utilidad

No son suficientes las metas, solas, para generar una conducta de gran calidad. Por ejemplo, para llegar de un punto a otro KITT lo puede lograr de diferentes formas: pasar por Chapultepec aunque se vaya a Iztapalapa (más corta vs. tráfico), yendo a más de 100 km x hora (¿segura?), dando arrancones (¿más gasolina?), no permitiendo que nadie se atraviese (¿confiable?), etc.

La utilidad (calidad de ser útil) se considera como una función que mapea de un estado (o secuencias de estados) a un número real.

Las metas tienen que ver con estados "contento" (cumplido?) y "no contento" (no cumplido?) mientras que una medida de desempeño más general debería permitir una comparación de diferentes estados del mundo (o secuencias de estados). Entonces si se prefiere un estado a otro existe una mayor utilidad de un agente.



Agente basado en utilidad

La especificación de la utilidad permite decisiones racionales cuando las metas presentan problemas: 1) cuando existen metas con conflictos (velocidad vs. seguridad) la función de utilidad especifica el compromiso adecuado por el que se puede optar; 2) cuando hay varias metas que el agente puede desear obtener, ninguna se puede lograr con certeza, la utilidad provee una forma por la cuál la posibilidad de éxito puede ponderarse contra la importancia de las metas.

Para R & N un agente racional puede describirse poseyendo una función de utilidad. Un agente que posee una función de utilidad explícita puede hacer decisiones racionales pero tiene que comparar las utilidades logradas por diferentes cursos de acciones. Las metas, a pesar de su inflexibilidad, permiten al agente tomar inmediatamente una acción para satisfacer la meta. Algunas veces se puede traducir una función de utilidad a un conjunto de metas, de tal forma que las decisiones hechas por el agente basado en metas empleando esas metas son idénticas (las decisiones) a las realizadas por el agente basado en utilidad.

EJERCICIO: cuál sería el mejor tipo de agente para el ejercicio anterior, explicar la selección.

3. BÚSQUEDA

3.1. Solución de problemas

La solución de problemas en I A involucra los siguientes pasos: la definición del problema, el análisis del problema, la representación del conocimiento requerido, y la elección del método adecuado de solución. Una característica de muchos de los problemas de IA es que la solución contempla tratar con una gran cantidad de datos. Existen diferentes métodos para encontrar las soluciones, una técnica clásica son los métodos de búsqueda. En estos métodos, se aplican reglas en combinación con una estrategia de control.

Por ejemplo, consideremos el problema de cripto aritmética siguiente:

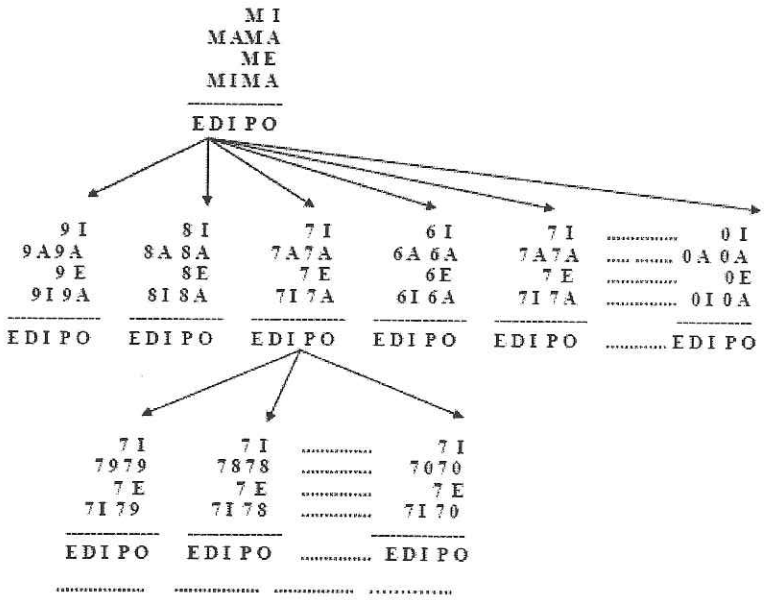
```
    M I
  M A M A
    M E
  M I M A
-----------
  E D I P O
```

7 letras distintas con distintos dígitos

604 800 combinaciones

Para este ejemplo, la regla a utilizar asigna un posible valor a la vez a una letra específica, que sea diferente de los valores asignados a las otras letras. En este caso la estrategia de control, es asignar los valores sucesivos de 9 a 0. Cuando existen varias reglas la estrategia de control especifica el orden en el que se deben aplicar las reglas y la forma de resolver conflictos.

En este ejemplo para el sistema decimal, los estados podrían ser miles si consideramos que la primera letra puede tener hasta diez valores posibles, la siguiente nueve, etc. La solución se encontrará cuando en algún momento aparezca un estado donde todas las letras tienen un valor específico y satisfacen la suma indicada.



Descripción del problema

En los métodos de búsqueda una forma de describir los problemas es usando *árboles*, donde se identifican los siguientes elementos:

1) nodo inicial ***i***
   Es el estado inicial del problema, consiste en uno o varios estados en los que comienza el problema
2) nodo final ***g***
   Es la meta, consiste en uno o varios estados finales que se consideran una solución al problema
3) procedimiento u operador (reglas y control que se aplican para pasar de un estado a otro)
   El procedimiento sucesor (o función), encuentra los sucesores dado un nodo
   El operador denota la descripción de una acción en términos de cuál estado se alcanzará llevando a cabo la acción en un estado particular.

   Notar que procedimiento u operador permite construir el árbol conforme avanza el proceso

En los métodos de búsqueda, el espacio de búsqueda corresponde a todas las posibles trayectorias, aunque no las tenga almacenadas. El espacio de estados corresponde a todos los estados alcanzables desde el estado inicial.

La solución en un método de búsqueda es una trayectoria del nodo inicio a la meta, cuando se quiere saber cómo llegar a la solución. Si la solución es lo único importante no es necesario almacenar las trayectorias de todos los nodos.

Regresando al problema de cripto aritmética, podemos describir el problema como:

Nodo inicial – la suma de los operandos: MI, MAMA, ME, MIMA, dando como resultado EDIPO

Nodo final – el rompecabezas de Cripto aritmética contiene solamente dígitos, y representa una suma correcta

Función de nodos sucesores – reemplazar todas las ocurrencias de una letra con un dígito que no aparezca ya en el rompecabezas

Un estado es la representación del problema en un instante dado, en este ejemplo un estado es el rompecabezas de Cripto aritmética con algunas letras reemplazadas por dígitos. El espacio de estados son todos los posibles rompecabezas que se puedan generar con letras reemplazadas por dígitos.

EJERCICIOS. Método de Búsqueda:

1) Describa el problema de ir de CU al zócalo
2) Describa el problema de planificar las actividades de un día

Análisis del problema

Para elegir el método que resuelva el problema es necesario extraer información sobre características de la solución, sobre cuestiones que afectan la definición de la solución, sobre el tamaño del espacio de estados, etc.

Regresando otra vez al ejemplo de criptoaritmética, tenemos la siguiente información:

- No es importante la trayectoria del nodo inicio a la meta, la solución solamente requiere los valores asignados a cada letra
- El conocimiento del problema (sistema decimal, operador suma) puede utilizarse para restringir la búsqueda de la solución.

- No es necesario encontrar la mejor solución, si hay varias soluciones, con encontrar una solución es suficiente

Esta información, ayudará a definir la complejidad del problema. Si fuera un problema donde la trayectoria es parte de la solución, deberá considerarse que la solución del problema requiere mantener en memoria las trayectorias de todos los nodos. Si el espacio de estados es muy grande y el problema lo permite, se podría descomponer el problema en subproblemas independientes. Si el entorno es estático, es posible planificar la secuencia de operadores que garanticen llegar a la solución, etc.

Es importante notar que para un número específico de estados en un problema dado, puede haber un número infinito de trayectorias en el árbol de búsqueda, es decir, un número infinito de nodos en el árbol de búsqueda. Esto ocurre aún si no hay ciclos.

3.2. Elección del método para resolver el problema

En este apartado nos referimos a la elección del método. La representación del conocimiento requerido se tratará en el capítulo representación del Conocimiento. Cuando no se tiene un método para obtener un resultado de forma directa, los métodos de búsqueda resultan fundamentales en la resolución del problema. A continuación se presenta el algoritmo general de los métodos de búsqueda y enseguida las variaciones más empleadas.

Métodos de búsqueda: solución general

Al describir los problemas en forma de árbol, la solución o las soluciones del problema se encuentran recorriendo dichos árboles. Las dimensiones de los árboles inciden en la complejidad computacional de las soluciones. Los parámetros esenciales son: ramas y profundidad.

A continuación, presentamos el algoritmo general que recorre un árbol de estados cuya raíz es el estado inicial, y en cada nivel se hallan los estados sucesores correspondientes.

1) Lista *L* de nodos iniciales. En cada paso *L* tiene los nodos sin examinar.
2) Si *L* está vacío → falla, si no ***tomar*** un nodo *n* de *L*
3) Si *n* es un nodo meta, parar y regresar el nodo con su trayectoria
4) Si no, quitar *n* de *L* y ***añadir*** a *L* todos sus hijos con su trayectoria completa.
5) Regresar a 2)

Prácticamente las variantes de este algoritmo difieren en dos puntos: cómo tomar el siguiente nodo y cómo añadir los nodos sucesores. Una clasificación basada en cómo tomar los nodos divide a los métodos en:

- Métodos de búsqueda ciega, cuando se toma siempre el primer nodo en la lista
- Métodos de búsqueda informada, donde existe alguna métrica para evaluar los nodos, se toma un nodo de acuerdo a ese valor

Considerando esta clasificación, en la búsqueda ciega tiene sentido hacer diferencia en cómo añadir los nodos sucesores a la lista, ya que siempre se tomará el primero. En cambio, en la búsqueda informada, no se requiere especificar cómo añadir los nodos sucesores a la lista, ya que se evaluarán de acuerdo a alguna métrica y ese valor definirá cuál nodo se toma primero.

El número de nodos sucesores define el número de ramas y el procedimiento u operador define la profundidad del árbol, es decir, cuántos niveles tendrá el árbol. En los métodos de búsqueda se considera que el número de ramas es finito, de otra forma no sería posible asegurar ni el primer nivel de nodos sucesores.

Hay dos costos a considerar en los problemas cuya solución se basa en métodos de búsqueda:

- Costo de búsqueda – el costo de encontrar una trayectoria del inicio a la meta, es decir, de encontrar una solución
- Costo de la solución – el costo de emplear la solución encontrada

Generalmente, un costo alto de búsqueda implica una solución de menor costo, es decir, mientras más se tarda la búsqueda, mejor es la solución.

¿Cómo seleccionar el mejor método de búsqueda para un problema dado?

Parámetros para establecer la eficiencia de cada método

- Espacio. ¿Cuánta memoria se usará en el método?
- Tiempo. ¿Cuánto tiempo tomará el método para encontrar una solución? (referido al costo de búsqueda)
- Completo. Cuando existe una solución, ¿el método la encontrará?
- Óptimo. Si existen varias soluciones, ¿ el método encontrará la mejor? (referido al costo de la solución)

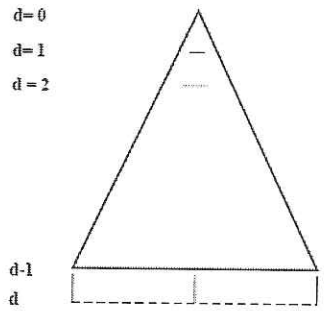
Respecto a los parámetros de tiempo y espacio, la complejidad se mide respectivamente en términos del número de nodos que se tienen que examinar (expandir) y del número de nodos que se mantienen en memoria. Sus valores se pueden calcular usando la información de número de ramas (usualmente referido como *b* 'branching') y el número de niveles de profundidad (usualmente referido como *d* 'depth').

3.3. Métodos de búsqueda ciega

Amplitud (Breadth-First Search)

Este algoritmo visita cada nodo del árbol por niveles, primero visita todos los nodos del primero, enseguida visita todos los nodos del segundo nivel, etc. Este método añade al final de L todos los hijos de *n*

1) Lista *L* de nodos iniciales
2) Si *L* está vacío → falla, si no tomar el primer nodo *n* de *L*
3) Si *n* es un nodo meta, parar y regresar el nodo con su trayectoria
4) Si no, quitar *n* de *L* y añadir al final de *L* todos sus hijos con su trayectoria completa.
5) Regresar a 2)



Considerando esta figura, los nodos expandidos se calculan de la siguiente forma para el peor caso. Notar que el factor *b* es constante, y la meta está en el nivel *d*:

$1 + b + b^2 + b^3 + ... + b^{d-1} = (b^d - 1) / (b-1)$ triángulo hasta (d-1)

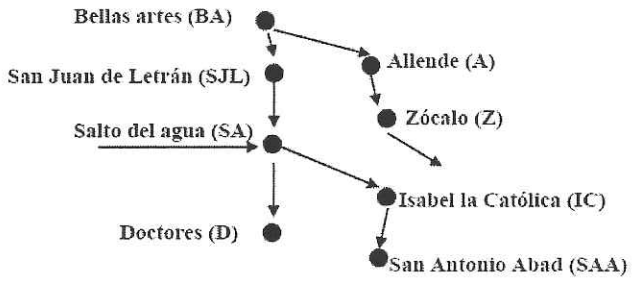
Para el último nivel se toma el promedio: mejor caso = 1, peor caso = b^d

Si d >> b la complejidad es $O(b^d)$

Si b=10 y d=5 el número de expansiones será de 111,111

Algoritmo de búsqueda en Amplitud

En el siguiente ejemplo del metro ya están expandidos los nodos de un área. Se quiere ir de Bellas Artes a San Antonio Abad.



Inicio	BA	Expandir: BA por SJL, A
1	SJL | A	Expandir: SJL por SA
2	A | SA	Expandir: A por Z
3	SA | Z	Expandir: SA por D, IC
4	Z | D | IC → D | IC	No se puede expandir: Z
5	D | IC → IC	No se puede expandir: D
	SAA	Expandir: IC por SAA

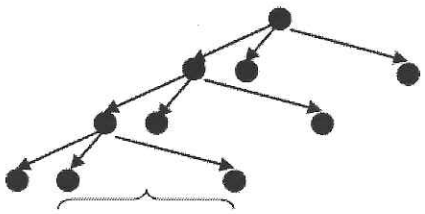
El método de amplitud trabaja bien incluso en árboles con profundidad infinita. Pero resulta malo si el número de ramas es grande.

Tiempo	Espacio	¿Completo?	¿Óptimo?
$O(b^d)$	$O(b^d)$	Si	Si

Profundidad (depth)

Este algoritmo visita cada nodo del lado izquierdo del árbol hasta llegar al último nivel y regresa por las siguientes ramas desde los niveles más profundos. Este método añade al frente de L todos los hijos de *n*

1) Lista *L* de nodos iniciales
2) Si *L* está vacío → falla, si no tomar el primer nodo *n* de *L*
3) Si *n* es un nodo meta, parar y regresar el nodo con su trayectoria
4) Si no, quitar *n* de *L* y añadir al frente de *L* todos sus hijos con su trayectoria completa.
5) Regresar a 2)



Considerando la figura, los nodos expandidos se calculan:

Espacio: $d(b-1)+1$

Tiempo: mejor caso = $(d+1)$, peor caso = $1 + b + b^2 + \ldots + b^d = (b^{d+1} - 1) / (b-1)$

Promedio: $[\,(d+1)(b-1) + (b^{d+1} - 1)\,] \;/\; 2(b-1)$

si $d >> b$ $O(b^d)$

Considerando el ejemplo anterior:

Inicio	BA	Expandir: BA por SJL, A
1	SJL | A	Expandir: SJL por SA
2	SA | Z	Expandir: SA por D, IC
3	D | IC | A	No se puede expandir: D
5	IC | A	Expandir: IC por SAA
	SAA | A	

Tiempo	Espacio	¿Completo?	¿Óptimo?
$O(b^d)$	O(bd)	Si (prof. finita)	No

El método de amplitud es ligeramente más ineficiente en tiempo y bastante más ineficiente en espacio que el método de profundidad.

Complejidad en tiempo y espacio del método de amplitud, se asume que b = 10; se procesan 10^6 nodos/segundo; se emplean 100bytes/nodo

d	#Nodos	Tiempo	Memoria
2	111	.01 mseg	11 Kbytes
4	11,111	1 mseg	1 Mbyte
6	$\sim 10^6$	1 seg	100 Mb
8	$\sim 10^8$	100 seg	10 Gbytes
10	$\sim 10^{10}$	2.8 horas	1 Tbyte
12	$\sim 10^{12}$	11.6 días	100 Tbytes
14	$\sim 10^{14}$	3.2 años	10,000 Tb

3.4. Métodos iterativos

Estos métodos intentan mejorar la búsqueda de solución combinando las mejoras de los métodos de amplitud y profundidad. Se prefieren cuando hay un espacio de búsqueda muy grande y la profundidad de la solución es desconocida. Si se conoce la profundidad de la meta, se puede limitar la profundidad para hacer más eficiente la búsqueda. Sin embargo, generalmente es difícil definir previamente un límite adecuado para delimitar la búsqueda, por lo que los métodos iterativos son los adecuados.

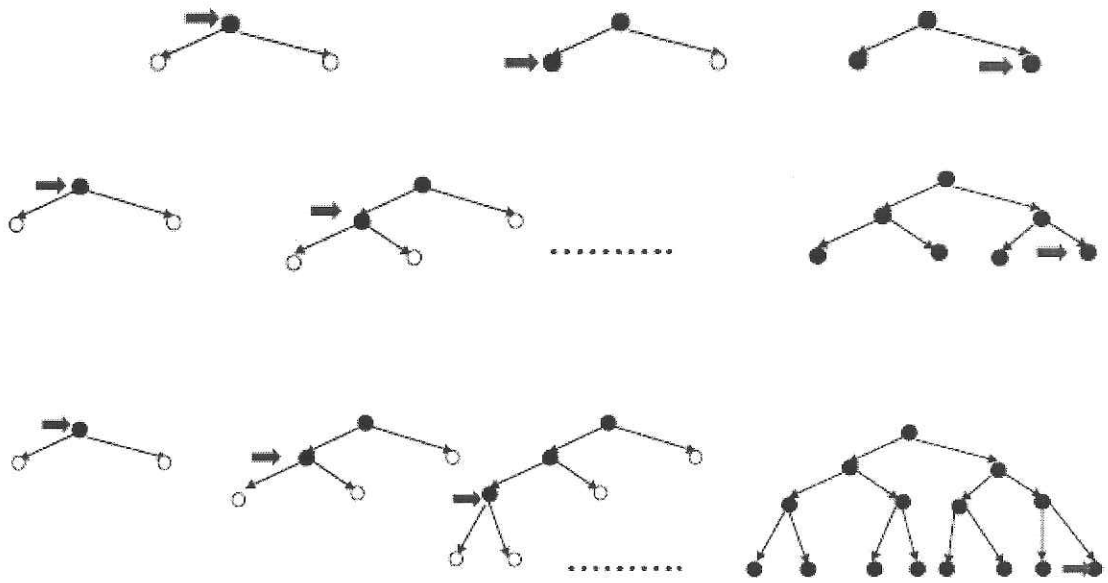
Profundidad iterativa (Iterative Deepening)

En el método de profundidad iterativa se añaden nodos al frente y se va aumentando la profundidad por iteraciones. De esta forma tiene la ventaja de espacio del método de profundidad y garantiza que se encuentre una meta de nivel mínimo, si existe la meta. En el método se van haciendo búsquedas con niveles limitados a 1,2,3, etc.

Este método ha servido para juegos de ajedrez con tiempo. El programa tiene un tiempo límite para seleccionar su decisión. De esta forma, puede hacer una búsqueda de nivel 2, luego de nivel, 3, después de nivel 4, y así sucesivamente dentro del tiempo establecido y regresa la mejor posibilidad.

Profundidad Iterativa

1) Fijar C=1 como profundidad inicial
2) Lista *L* de nodos iniciales
3) Sea *n* el primer nodo de *L*
   Si *L* está vacío → incrementar C en 1 y regresar a 2
4) Si *n* es un nodo meta, parar y regresar el nodo *n* con su trayectoria
5) Si no, quitar *n* de *L*
   Si la profundidad de *n* es menor que C añadir al frente de *L* todos sus hijos con su trayectoria completa.
6) Regresar a (3)



Tiempo	Espacio	¿Completo?	¿Óptimo?
$O(b^d)$	O(bd)	Si	Si

El número de nodos expandidos por este método no es mucho mayor que en el método de amplitud. Para calcular el número de nodos expandidos, notamos que se expande el mismo número de nodos que en el método de amplitud a una profundidad *d*, pero que en el peor caso, con la meta en *d*, tiene que repetir búsquedas, así que el cálculo es el siguiente:

$(d+1)1 + (d)b + (d-1)b^2 + \ldots + b^d$ expansiones

123,456 nodos si b=10 y d=5, que corresponde a un aumento de 11% sobre Amplitud

Amplitud iterativa (Iterative broadening)

En el método de amplitud iterativa se añaden nodos al frente y se va aumentando el número de ramas por iteraciones. Tiene la ventaja del espacio del método de profundidad y trata de encontrar más rápido la meta saltando ramas. Se van haciendo búsquedas limitando el número de ramas a 1, 2, 3, etc.

Consideraciones

El algoritmo apropiado depende del problema a solucionar. Por ejemplo, la búsqueda en profundidad es un buen método si se cree que todas las trayectorias parciales llegan a callejones sin salida o se vuelven soluciones completas a profundidades razonables.

Si se busca el camino más corto, el método de búsqueda en amplitud puede ser el mejor.

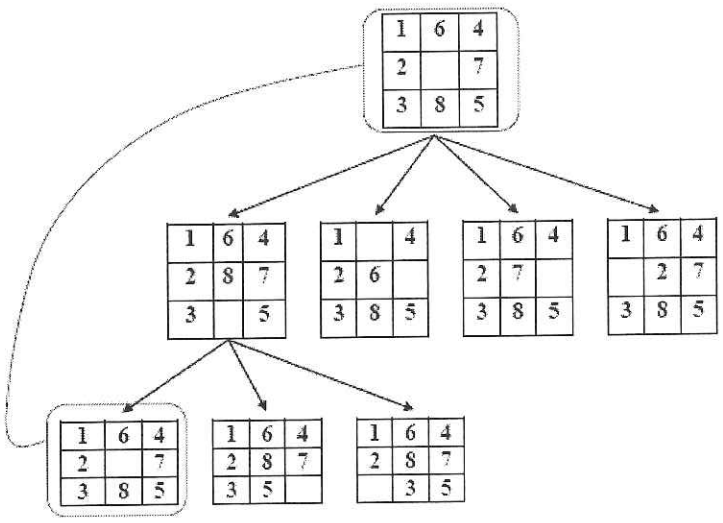
Si la memoria es un problema, el método de profundidad generalmente requiere mucho menos memoria.

Si se quiere encontrar rápidamente una solución, considerar:

- Profundidad puede ser más rápido si hay muchos caminos que llevan a estados meta pero todos son muy largos.
- Búsqueda en amplitud puede ser más rápido si hay un camino corto a la meta, pero dentro de un espacio de búsqueda grande y profundo.

3.5. Estados repetidos

Un problema que no se ha considerado en los ejemplos anteriores es el caso de nodos repetidos, es decir, la expansión de nodos previamente expandidos, y que provocan desperdicio de recursos en el proceso de búsqueda. Un ejemplo es el caso del juego de las 8 placas, donde hay pasos reversibles:



Soluciones a estados repetidos

Requiere comparar las descripciones de estados. ¿Cómo podemos impedir que aparezcan estados repetidos?

1) Mantener rastro de todos los nodos generados.
2) Nunca extender una trayectoria a un estado del cual ya venía (estado previo). Esta opción es la más simple.
3) Evitar trayectorias con ciclos, comprobando si un nuevo estado ya ha aparecido en cualquier punto de la trayectoria actual. Evita ciclos dentro de trayectorias individuales, pero no la posibilidad de repetición de estados en otras trayectorias.
4) No generar ningún estado que ya ha sido visitado, independientemente de la trayectoria. Esta opción puede no ser deseable, consideremos el problema de encontrar una ruta ya que se puede ir del estado X al Y o del Y al X.

3.6. Búsqueda Heurística

Los métodos de búsqueda ciega son simples pero también imprácticos muchas veces. No tienen ninguna información sobre el espacio de estados que les permita decidir en cada estado, exploran en una forma sistemática. Los métodos de búsqueda heurística intentan mejorar la eficiencia, explorando las mejores trayectorias según una valoración del estado.

> Heurística [Del Griego εὑρίσκειν, hallar, descubrir] http://buscon.rae.es/draeI/
> **1.** adj. Perteneciente o relativo a la heurística.

2. f. Técnica de la indagación y del descubrimiento.
3. f. Busca o investigación de documentos o fuentes históricas.
4. f. En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

Los métodos informados añaden información específica de dominio para seleccionar la mejor trayectoria conforme se va buscando. Se define una función heurística, h (n), que estima "qué tan bueno" es un nodo n. Específicamente, h (n) evalúa cada nodo respecto a la meta. La función heurística es una estimación, basada en la información específica de dominio, que puede calcular a partir de la descripción del estado actual, qué tan cerca está de la meta.

Heurística

Todo el conocimiento de dominio que se usa en la búsqueda se codifica en la función heurística h. En general:

h (n)> = 0 para todos los nodos n
h (n) = 0 implica que n es un nodo meta
h (n) = ∞ implica que n es un callejón sin salida del cual no puede alcanzarse la meta

El término "débil" para un método se refiere a métodos que son muy generales y no adaptados a una situación específica. Se llaman métodos "débiles" porque no aprovechan heurísticas más poderosas de dominio específico.

Para usar la búsqueda heurística se necesita una función de evaluación que dé un puntaje a cada nodo en un árbol de búsqueda según qué tan cerca de la meta parezca estar. Aunque pueda ser solamente una estimación no precisa, evita seguir todas las posibles trayectorias, limitándose a las trayectorias que parecen estar más próximas a la meta

Por ejemplo, en el espacio de búsqueda del metro del D. F. la distancia en línea recta podría usarse como heurística, favoreciendo las trayectorias donde los nodos estén más cerca de San Antonio Abad. Aunque las heurísticas no son precisas e incluso pueden no ser confiables en algunos casos, proveen un ordenamiento de exploración que disminuye el espacio de búsqueda, generalmente ayudan a encontrar una solución más rápido. El ordenamiento se debe a que la función de evaluación clasifica los nodos y ya no es importante insertar los nodos en la lista en un orden específico.

Función Heurística

La función h (N) estima el costo de la trayectoria más barata desde el nodo N al nodo meta. Ejemplo: del juego de 8 placas:

1	6	4
2		7
3	8	5

1	2	3
8		4
7	6	5

h1 (N) = el número de placas desacomodadas = 7 es admisible

h2 (N) = la suma de distancias de cada placa a la meta = 18 es admisible

Sobre Heurística

- La heurística se quiere para orientar la búsqueda a lo largo de caminos prometedores
- El tiempo gastado en calcular la heurística debe recuperarse en una mejor búsqueda
- Una función heurística puede ayudar a solucionar el problema; dirigiendo perfectamente la búsqueda
- La decisión de qué nodo expandir se llama, algunas veces, meta-razonamiento
- La heurística no siempre puede traducirse a números y puede implicar una cantidad grande de conocimiento

Primero por lo mejor

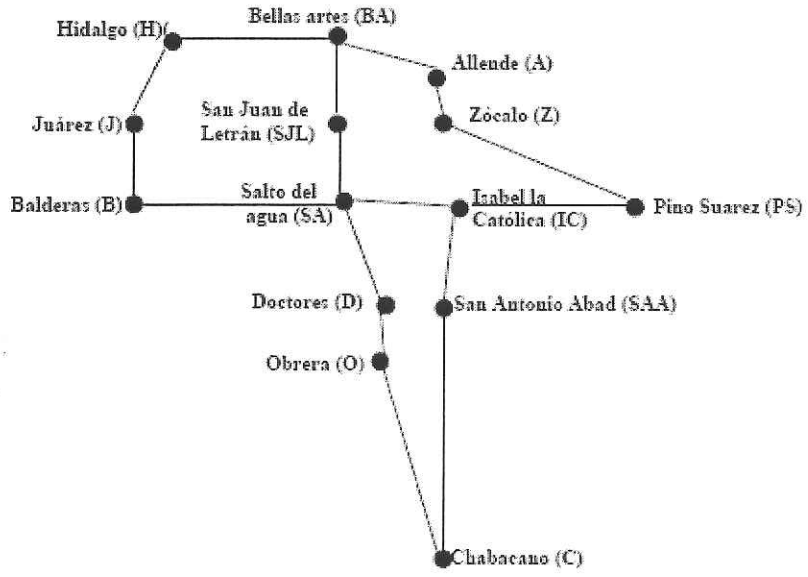
Es una forma genérica de referirse a los métodos informados. Los nodos se clasifican de acuerdo al valor dado por la heurística.

1) **Lista *L* de nodos iniciales**
2) **Sea *n* el nodo de *L* que se espera sea el más próximo a la meta**
3) **Si *L* está vacío → falla**
4) **Si *n* es un nodo meta, parar y regresar el nodo con su trayectoria**
5) **Si no, quitar *n* de *L* y añadir a *L* todos los hijos de *n* con su trayectoria completa.**
6) **Regresar a 2)**

Búsqueda avara

La búsqueda avara usa como función de evaluación la heurística f (n) = h (n). Selecciona el nodo que se cree más cercano a la meta como el nodo a expandir, es decir, el nodo con el valor más pequeño de h.

Si consideramos el ejemplo siguiente y la heurística de distancia en línea recta, veremos que el método no es completo.



Por ejemplo, si se quiere ir de Pino Suárez a Doctores:

Inicio	PS	Expandir: PS por Z, IC
1	IC, Z	Expandir: IC por SA, SAA, PS
2	SAA, SA, Z, PS	Expandir: SAA por C, IC
3	IC, SA, Z, PS, C	Expandir: IC por SA, SAA, PS

Si el espacio de estados es finito y se evitan los estados repetidos, la búsqueda avara puede ser completa.

El método no es óptimo en general, tomemos el ejemplo de ir de Bellas Artes a Pino Suárez, con la heurística de distancia en línea recta tomaría la trayectoria BA – A – Z – PS, como se trata del metro sabemos que no es un camino montañoso pero si fueran carreteras y el camino Z – PS fuera una cordillera la distancia sería mayor.

Tiempo	Espacio	¿Completo?	¿Óptimo?
$O(b^d)$ peor caso	$O(b^d)$	No	No

*Búsqueda A**

La búsqueda A* combina costo uniforme (el costo total mínimo desde el nodo raíz) y Avara. La función de evaluación es f(n) = g(n) + h (n), donde:

- g(n) es el valor del mejor camino encontrado hasta ahora a *n*
- h(n) es una heurística admisible
- f(n) es el valor estimado de la solución más barata a través de *n*

1) Lista *L* de nodos iniciales
2) Sea *n* el nodo de *L* para el cual f(n) = g(n) + h(n) es mínima
3) Si *L* está vacío → falla
4) Si *n* es un nodo meta, parar y regresar el nodo con su trayectoria
5) Si no, quitar *n* de *L* y añadir a *L* todos los hijos de *n* con su trayectoria completa.
6) Regresar a 2)

A: completo y óptimo*

Si hay una trayectoria del nodo inicial al nodo meta, A* (sin eliminar estados repetidos) será completo y óptimo, es decir, terminará por encontrar la mejor trayectoria, si:

1) La heurística es admisible y consistente
- la heurística h(n) es admisible si $0 \leq h(n) \leq h^*(n)$, donde h*(n) es el valor de la trayectoria óptima de n a la meta

- la heurística h(n) es consistente si para cada nodo *n* y cada sucesor *n'* de *n*: $h(n) \leq c(n, n') + h(n')$. Si la heurística es consistente, entonces la función f es no decreciente $0 < \varepsilon \leq c(n, n')$

2) El espacio de búsqueda es una gráfica finita

Si h es consistente, siempre que A* expanda un nodo encontrará una trayectoria óptima para el estado asociado a ese nodo.

Por ejemplo en el juego de 8-placas

- h1(n) = número de placas desacomodadas
- h2(n) = suma de la distancia de cada placa a la meta

$h1(n) \leq h2(n) \leq h^*(n)$

Si h es consistente, entonces la función f(n) a través de cada trayectoria es no decreciente:

$f(n) = g(n) + h(n)$

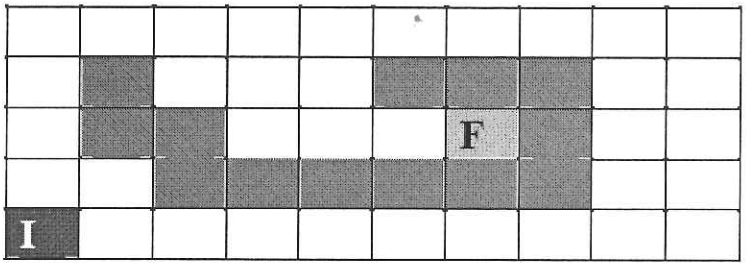
$f(n') = g(n) + c(n, n') + h(n')$

$h(n) \leq c(n, n') + h(n')$

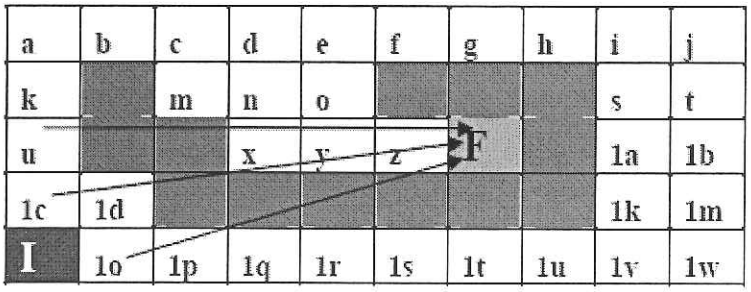
$f(n) \leq f(n')$

Ejemplo de Robot navegador

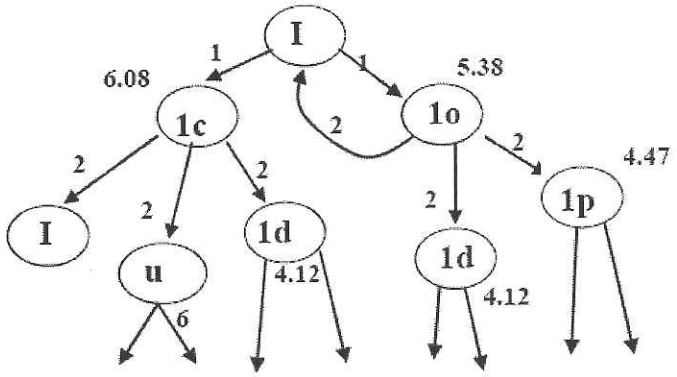
Consideremos un robot en el mundo de la figura siguiente donde el inicio es la celda I y la meta la celda F. El robot no puede avanzar en diagonal, solamente en forma horizontal y vertical. Las celdas grises indican paredes que no se pueden atravesar.



Como heurística usaremos la distancia en línea recta, donde el costo horizontal y vertical de un paso es 1. Las celdas están etiquetadas para identificarlas en el árbol de búsqueda.



En el siguiente árbol se muestra el espacio de búsqueda hasta el tercer nivel. Los números cerca de cada estado corresponden a la heurística, los números en las aristas corresponden a la función *g*.



*Evitar estados repetidos en A**

Si la heurística es consistente, entonces:

- La lista CERRADO tendrá la lista de estados asociados con nodos expandidos
- Cuando se genera un nuevo nodo:
  - Si el estado está en CERRADO, entonces eliminar n
  - Si tiene el mismo estado que otro nodo en el borde, entonces eliminar el nodo con el valor f más grande

Iterative Deepening A (IDA*) Ahondamiento Iterativo A**

- Usar f(n) = g (n) + h (n) con h admisible y consistente
- Cada iteración es búsqueda en profundidad con el límite en el valor de f de los nodos expandidos

1) Fijar C=1
2) Lista *L* de nodos iniciales. Sea C' = ∞
3) Sea *n* el primer nodo de *L*
   Si *L* está vacío y C' = ∞ → falla
   Si *L* está vacío, C' ≠ ∞ → C = C', regresar a (2)
4) Si *n* es un nodo meta, parar y regresar el nodo *n* con su trayectoria
5) Si no, quitar *n* de *L*; para cada hijo *n'* de *n*:
   Si f(n') ≤ C añadir *n* al frente de L
   Si no C' = min (C', f(n'))
6) Regresar a (3)

3.7. Búsqueda Local

Hill-Climbing (alpinismo de colinas, ascenso de colinas)

La función de evaluación se aplica a cada nuevo nodo para medir su cercanía al estado meta. Mientras más cerca de la meta está el nodo mejor será su resultado de evaluación. Sigue hacia el mejor nodo sucesor (y sólo si ese nodo es mejor que el actual).

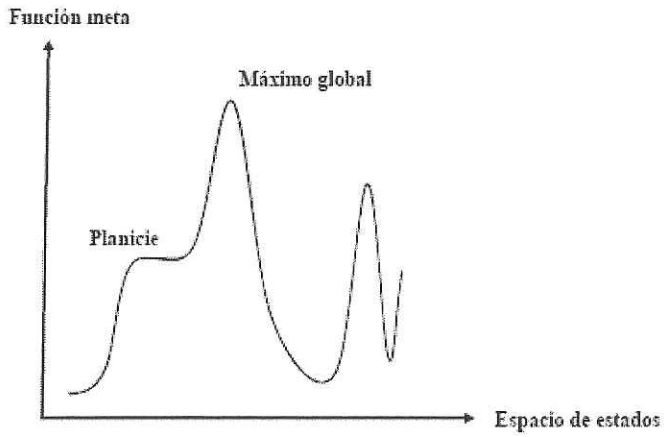
Este método trabaja haciendo cambios locales provisionales a su posición en el espacio de búsqueda. Sigue los movimientos que causan el cambio de estado más benéfico. Trabaja bien

cuando la meta se reconoce al instante. Sin embargo los problemas pueden ocurrir cuando la meta no está rigurosamente definida.

Por ejemplo, en un problema de sintonización de radio. Hay un control para sintonizar la frecuencia y la heurística subjetiva es la calidad del sonido. Alpinismo de colinas trabaja haciendo pequeños ajustes a la frecuencia y comprometiéndose al mejor.

Finalmente se alcanzará una calidad de sonido que es superior a la calidad de los estados vecinos. Se asume que se ha encontrado la mejor frecuencia así que se para la búsqueda.

Tiempo	Espacio	¿Completo?	¿Óptimo?
O(m)	Constante	No	No



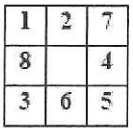
Si hay ciclos en el espacio de búsqueda entonces no se encontrarán las metas usando el alpinismo de colinas. El método simplemente para cuando no hay sucesores del estado actual que sean mejores que el estado actual. Esto puede llevar a problemas cuando se encuentran máximos locales en el espacio de búsqueda, es decir, puntos que son mejores que cualquier estado alrededor pero que no son la solución.

El alpinismo de colinas es apropiado para una clase limitada de problemas donde se tiene una función de evaluación que predice exactamente el costo estimado de una solución. Los problemas asociados con el alpinismo de colinas son:

- Problema de risco

El espacio de búsqueda puede contener riscos como estados de alta calidad que caen abruptamente a estados de baja calidad por cualquier lado. Puede no ser posible seguir exactamente las crestas angostas si los operadores de búsqueda no tienen suficiente detalle y el resultado puede ser una búsqueda oscilante.

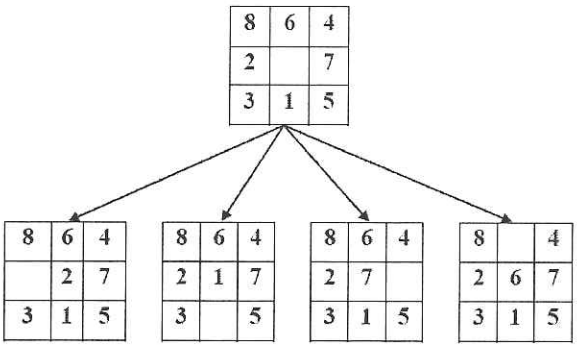
Por ejemplo, en el juego de 8 placas un estado es el siguiente. Con la heurística de placas desacomodadas su valor es 2, pero para poder sacar a 7 o a 3 para acomodarlas en su posición correcta es necesario ir a posiciones con peores valores.



- Problema de meseta

Si el espacio de búsqueda contiene una región grande plana (meseta) entonces los cambios locales no causarán ningún cambio en la función de calidad y el proceso de búsqueda no tendrá ningún modo de saber a dónde seguir.

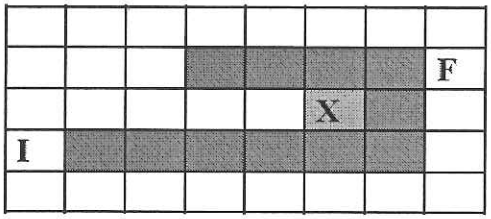
Por ejemplo, en el juego de 8 placas siguiente, con la heurística de placas desacomodadas, todos los nodos generados tienen el mismo valor.



- Problema de máximo local

A menudo hay picos en el espacio de búsqueda que representan máximos locales. El proceso de búsqueda puede alcanzar un punto donde los cambios locales sólo degradan el valor de calidad y entonces la búsqueda puede pararse conforme a la suposición de que un estado óptimo ha sido encontrado, cuando realmente sólo se han encontrado unos máximos locales.

Por ejemplo, en el laberinto siguiente, X es un máximo local y es necesario retroceder para llegar a F.



¿Cómo solucionar estos problemas?

- Reinicio aleatorio, después de un número predefinido de pasos locales
- Recocido simulado
- Búsqueda Tabú, previene estados repetidos al tener una lista de longitud fija máxima y si un nodo ya se encuentran en ella no se considera.
- Otros (programación de algoritmos Genéticos...)

Búsqueda en haz (Beam search)

La búsqueda en haz se parece al de amplitud porque busca por niveles

Evita la complejidad espacial de amplitud limitando el número de nodos que expande en cada nivel. Sólo expande los *x* nodos más prometedores (x es el número de ramas, el haz).

Se usa una función heurística para juzgar cuáles son los mejores nodos *x* para expandir en un nivel particular.

Usa una función de evaluación f (n) = h (n), pero el tamaño máximo de la lista de nodos es k, una constante fija. Sólo guarda los *K* mejores nodos como candidatos por expansión, y tira el resto

Más espacio eficiente que en la búsqueda avara, pero puede desechar un nodo que está en un camino de solución

- No completo
- No óptimo

Simulated annealing

Búsqueda: expresada como optimización de función

Muchos problemas de la IA son una forma de optimización de función. Las salidas de estas funciones es un valor que indica lo deseable de un estado, definido por las entradas de la función.

Considerando el TSP (traveling sales problem) distancia mínima de la ruta.

Cada tour es un estado en el espacio de búsqueda. Para el TSP, la función que se evalúa es la distancia correspondiente al estado. El truco es generar los estados durante la búsqueda.

Si todas las combinaciones de trayectorias (saltillo, banderillas, lerma, nautla, cocula)

SBLNCS 602

SBLCNS 569

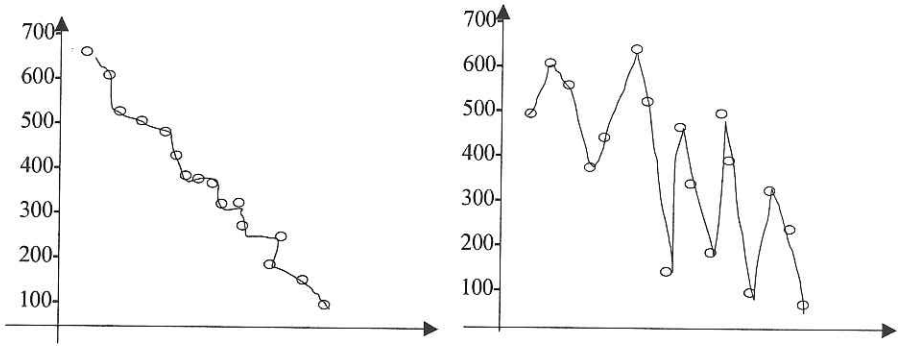
SBCLNS 591

SBCNLS 528

....

De 400 a 650 km.

Si se grafican, clasificadas en orden descendente, cada punto es un tour, se unen solamente para un impacto visual, pero realmente la función es discreta. El espacio de búsqueda en la práctica aparecería como en la segunda figura. El eje horizontal denota tiempo.



La búsqueda sobre todas las curvas (que forman un valle) es la búsqueda de pesos que proveen el mejor ajuste. Pero en muchos problemas esta superficie es compleja con picos y valles. La tarea de optimización es encontrar el punto más bajo en ese paisaje.

Imaginar una maqueta de un terreno montañoso con muchos picos y valles. Si tiramos una pelotita en él, ésta se moverá entre valles y colinas hasta quedar en un hoyo. La parte más profunda de la maqueta es un mínimo global, el punto mínimo de una región es un mínimo local.

El terreno montañoso se puede pensar como una función de costo de dos parámetros que describen las coordenadas donde la pelota va brincando y finalmente reposa. La solución ideal es donde el costo es un mínimo global.

En muchos problemas reales, la superficie de la función de costo es demasiado grande para explorarla exhaustivamente. El algoritmo que actúe como una bola de ping-pong, con cierto control como lo hace la pelota es "Simulated annealing".

Recocido (templado) simulado se usa recientemente para enfrentar numerosos problemas de optimización. El templado de metales se hace para endurecer el material. El material se calienta para que los átomos tengan suficiente energía para moverse. Si el material se enfría muy pronto los átomos se paralizan en orientaciones arbitrarias, se enfría con un horario diseñado para que los átomos del material logren estados de baja energía y se pongan en fila adecuadamente.

Este algoritmo se puede utilizar para el problema de TSP:

1) empieza con una lista seleccionada al azar: SNBLCS
2) se necesita un método para generar un estado nuevo (tour nuevo) del estado existente y un método para calcular el cambio de energía.
3) El costo es la longitud del tour.
4) Un nuevo estado se puede obtener seleccionando dos ciudades e intercambiando su posición: SNCLBS (C y B se escogen)
5) La energía se calcula como la longitud del tour
6) Si el nuevo estado genera una reducción en la longitud del tour (o permanece igual) se acepta como el nuevo tour
7) Si el nuevo estado genera un tour más largo entonces se selecciona como nodo actual si $p' = e^{\text{delta E}/kT}$ es un número entre 0 y 1

Existen problemas en los cuales no se tiene un objetivo definido explícitamente sino una función 'v' definida sobre una estructura de datos, siendo el objetivo del proceso de búsqueda localizar una estructura que maximice (o minimice) la función.

Si consideramos que la estructura de datos son 'puntos' en un espacio, la función se puede ver como un paisaje definido sobre ese espacio.

Así, podemos decir que un tipo de métodos de optimización está constituido por aquellos métodos que recorren el paisaje en busca de los puntos situados con mayor altura. Sin embargo, no aseguran encontrar el máximo global, ya que en la mayoría de los casos no es conocido.

4. Juegos y su solución mediante búsqueda

Los juegos han provocado fascinación y desde que existen las computadoras, se ha intentado que las computadores puedan jugar. Turing en 1951 diseñó las instrucciones para una máquina capaz de jugar ajedrez. Shannon en 1950 publicó "Programming a Computer for Playing Chess"[4].

Hasta este punto se han considerado ambientes de un solo agente. Los juegos son una forma de ambiente de varios jugadores. El ajedrez sería un juego de dos agentes cuando hay dos jugadores únicamente o multiagente cuando se trata de un jugador y varios contrincantes. En los ambientes multiagentes las acciones de cada uno pueden afectar las decisiones de los otros agentes y sus resultados. Los ambientes multi-agente se clasifican en cooperativos y competitivos. Los competitivos dan origen a problemas de juegos de adversarios.

Los métodos de búsqueda también se usan para la solución de juegos. En la aplicación de búsqueda para un solo agente, la solución es aplicar el método para encontrar la meta y las heurísticas y otras técnicas ayudan a encontrar la solución óptima. En la aplicación de búsqueda para solucionar juegos de varios participantes (de adversarios), la solución es una estrategia y ésta especifica una movida para cada posible respuesta del oponente. La función de evaluación da un valor de qué tan buena es una posición en el juego.

Para analizar la complejidad de los juegos Norvig y Russell consideran los parámetros de información (perfecta e imperfecta) y su determinismo (o elementos aleatorios). Si la información es perfecta, todo está a la vista de los participantes, si es imperfecta hay algo desconocido para ellos. Algunos ejemplos de acuerdo a estas características se presentan a continuación:

	Determinístico	**Aleatorio**
Información perfecta	ajedrez, damas, go	monopolio, backgammon
Información imperfecta	battleships	poker, bridge, scrabble

4.1. Juegos de dos adversarios

En juegos de dos personas el algoritmo más empleado es MINIMAX. Un jugador es MAX y otro jugador es MIN. MAX mueve primero y el siguiente turno es para MIN, así que se van alternando en los niveles del árbol de búsqueda.

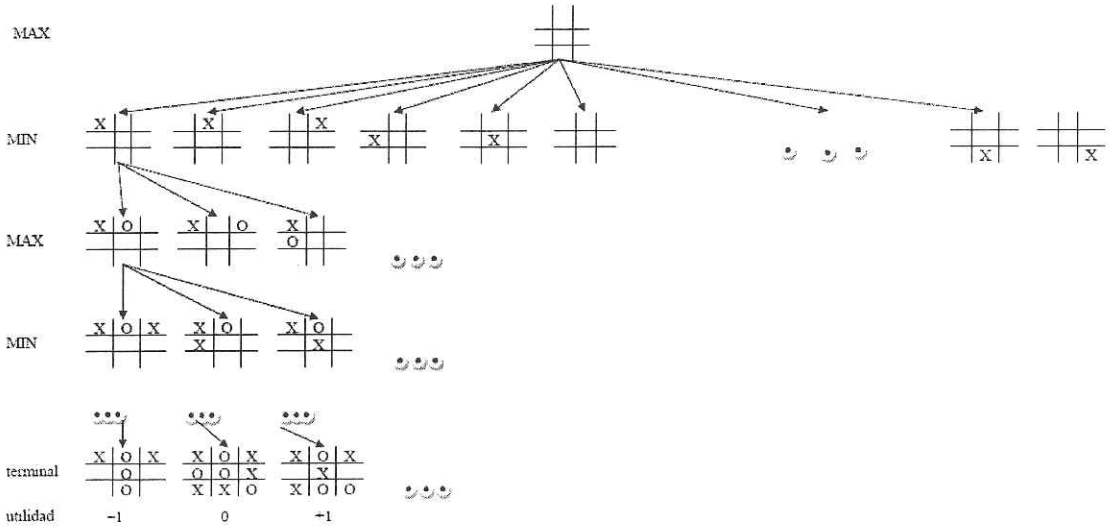
En la búsqueda de adversarios:

- Nodo inicio (o nodos inicio)
- Función de nodos sucesores, especifica los movimientos legales
- Prueba terminal, para saber si ya terminó el juego
- Función de utilidad, da un valor a cada estado terminal. Por ejemplo: gana (+1), pierde (-1), empata (0)

En el juego de adversarios se consideran las siguientes suposiciones: 1) MAX asume que MIN es un adversario infalible, 2) MAX y MIN juegan óptimamente

Ejemplo de un árbol de búsqueda para el juego de gato. MAX utiliza el árbol para determinar sus movimientos.

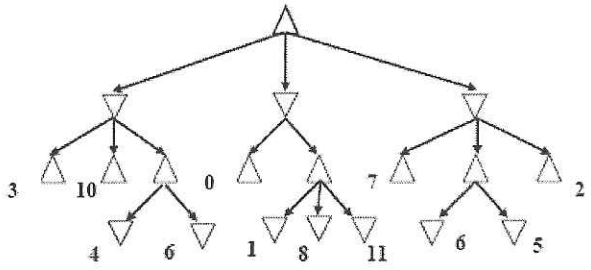
[4] http://en.wikipedia.org/wiki/Computer_chess



4.2. Algoritmo Minimax

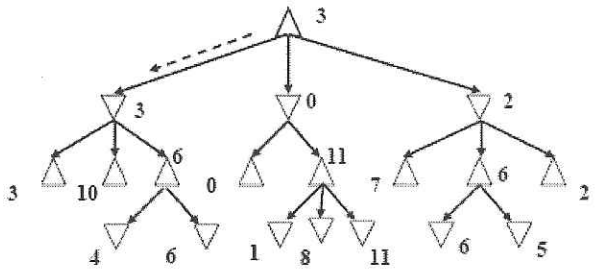
1) Sea *N* la lista consistiendo de *m* (raíz)
2) Sea *n* el primer nodo de *N*
3) Si *n* = *m* y *n* tiene asignado un valor, salir regresando este valor
4) Si *n* tiene un valor asignado, eliminar *n* de *N*
5) Si *n* no tiene valor asignado y es un nodo terminal, asignar a *n* el valor de +1, 0 o -1 dependiendo de si gana, empata o pierde
6) Si *n* no tiene valor asignado y todos sus hijos tienen valores asignados, asignar a *n* el valor máximo de los valores de sus hijos si es un nodo MAX, en caso contrario asignar el mínimo de los valores de sus hijos
7) Si *n* no tiene valor asignado y todos sus hijos no tienen valores asignados, añadir los hijos de *n* al frente de N
8) Regresar a (2)

Supongamos un juego que tiene el siguiente árbol:



En los nodos hojas se muestran los valores de acuerdo al punto 5 del algoritmo, en este ejemplo los valores no son de suma cero[5] sino en un rango amplio. El triángulo Δ indica un nodo MAX y hacia abajo se trata de un nodo MIN

Aplicando el algoritmo minimax se obtienen los valores mostrados en la siguiente figura, donde la línea punteada indica la decisión final de MAX para el juego indicado.



MINIMAX es una búsqueda en profundidad, así que su eficiencia es igual, a continuación *m* indica la profundidad del nodo meta.

Tiempo	Espacio	¿Completo?	¿Óptimo?
$O(b^m)$	O(bm)	Si	Si (finito)

Desventajas de MINIMAX

- El número de estados es exponencial al número de movimientos
- Genera todo el árbol hasta alcanzar estados terminales
- Usa los valores de los estados terminales para calcular el valor de los nodos superiores hasta llegar a la raíz

Función de evaluación

Cuando se trata de espacios de búsqueda muy grandes, las limitaciones de recursos deben considerarse. Si es posible buscar mil nodos por segundo, en cien segundos se podrían obtener 10^5 nodos, que sigue siendo un espacio de búsqueda muy grande.

Una solución es controlar la cantidad de búsqueda mediante una función de evaluación que estime las posibilidades de ganar de cada jugador, es decir, la utilidad esperada del juego dada una posición. Esta evaluación se basa en características fácilmente calculables en cada estado. Esta evaluación debe coincidir con la función de utilidad en los estados terminales. El desempeño dependerá de la calidad de la función de evaluación. Su cálculo no debe tomar demasiado tiempo.

Muchas funciones de evaluación se especifican como suma ponderada de características:

$$(w_1 * caract_1) + (w_2 * caract_2) + \ldots + (w_n * caract_n)$$

Por ejemplo en el juego de ajedrez algunas características evalúan la posición de la pieza en el tablero y otras características describen las configuraciones de varias piezas. Por ejemplo: $w_1 = 9$ con $caract_1$ = diferencia de reinas (# reina blanca - # reina negra).

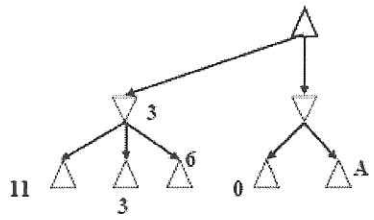
---

[5] La ganancia o pérdida de un participante se equilibra con exactitud con las pérdidas o ganancias de los otros participantes

Esta solución puede complementar el corte de niveles. Por ejemplo, se puede generar el árbol hasta 7 niveles y aplicar la función de evaluación. El corte temprano de búsqueda sustituye la prueba-terminal por la prueba en un nivel límite. Se aplica la función de evaluación a los nodos de ese nivel. Establecer el más adecuado número de niveles es otro problema a solucionar.

4.3. Poda Alfa-beta

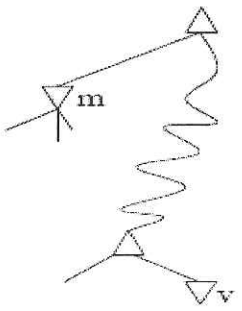
Otra solución es no examinar cada nodo, hacer algunas podas de ramas del árbol. Por ejemplo, en la siguiente figura, el algoritmo minimax evalúa el nodo Min de la izquierda. Al llegar al nodo 0 el segundo nodo Min ya tiene un valor de cero y podría podarse dependiendo de los valores de A.



Si $A \geq 0$ no importa ese valor porque el nodo Min escogerá el menor valor que es cero

Si $A \leq 0$ ese valor es importante para el nodo Min pero no para MAX-raíz ya que tiene una mejor opción en 3

En forma general, si m es mejor que el padre de v para el nodo MAX-raíz entonces se puede podar al padre de v y todos sus descendientes.



Cortes α

Si el valor MAX actual es mayor que el valor del sucesor MIN, ya se puede dejar de explorar el subárbol de MIN

Cortes β

Si el valor MIN actual es menor que el valor del sucesor MAX, se puede podar el árbol de MAX

Características de la poda α–β

La poda de subárboles completos no afecta el resultado final. Un buen ordenamiento de las movidas mejora la eficiencia del podado. Con un ordenamiento perfecto la complejidad en tiempo es de $O(b^{m/2})$

Alfa = el mejor valor encontrado para MAX a lo largo de su trayectoria

Beta = el mejor valor encontrado para MIN a lo largo de su trayectoria

Algoritmo Poda α - β

Sea *n* un nodo sobre el espacio de adversarios

1. Sea $L = n$

2. Sea *x* el primer nodo de *L*

   Si $x = n$ tiene un valor asignado, regresar este valor

3. Si *x* tiene un valor asignado (V_x)

   Sea *p* el nodo padre de *x*

   Verificación de podado

   a) Si *p* es uno nodo MIN

      Sea α = *max*{todos los valores actuales asignados a hermanos de *p* y a los nodos MIN antecesores de *p*}

      Si $V_x \leq \alpha$, eliminar *p* y todos sus descendientes de *L*

   b) Si *p* es uno nodo MAX

      Sea α = *min*{todos los valores actuales asignados a hermanos de *p* y a los nodos MAX antecesores de *p*}

      Si $V_x \geq \alpha$, eliminar *p* y todos sus descendientes de *L*

   Si *p* no puede podarse

   Sea V_p el valor actual asignado a *p*

   Si *p* es un nodo MIN, sea $V_p = \min\{V_p, V_x\}$

   Si no, si *p* es un nodo MAX, sea $V_p = \max\{V_p, V_x\}$

   Eliminar *x* de *L*

   Regresar a (2)

4. Si *x* no tiene valor asignado y es un nodo terminal, aplicar la función de evaluación.

Regresar a (2)

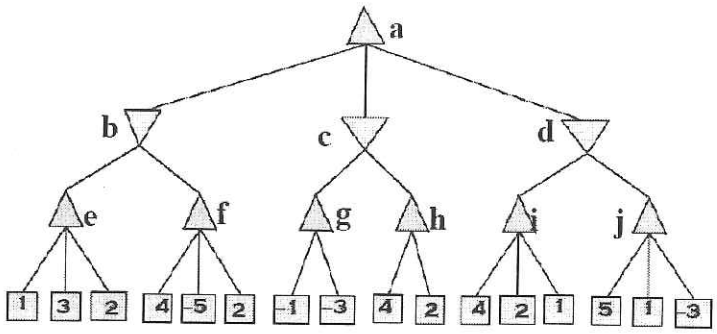
5. En otro caso

   Sea $V_x = -\infty$ si *x* es un nodo MAX y $V_x = +\infty$ si *x* es un nodo MIN

   Agregar todos los nodos hijos de *x* al frente de *L*

Regresar a (2)

Ejemplo:



Aplique la poda Alfa-Beta a la figura anterior

L = {a}, Va = $-\infty$

L = {b, c, d, a}, Vb = $+\infty$

L = {e, f, b, c, d, a}, Ve = $-\infty$

NP significa que p no pudo ser podado

L = {1, 3, −2, e, f, b, c, d, a}, Vx = 1, p = e, α= min{ϕ}, NP , Vp = max{$-\infty$, 1} = 1, eliminar nodo

L = {3, −2, e, f, b, c, d, a}, Vx = 3, p = e, α= min{ϕ}, NP , Vp = max{1,3} = 3, eliminar nodo

L = {−2, e, f, b, c, d, a}, Vx = −2, p = e, α= min{ϕ}, NP , Vp = max{3, −2} = 3, eliminar nodo, Ve = 3

L = {e, f, b, c, d, a}, x = e, Vx = 3, p = b, α= min{ϕ}, NP , Vp = min{+∞, 3} = 3, Vb = 3, eliminar nodo

L = {f, b, c, d, a}, x = f, Vp = $-\infty$

L = {4, −5, 2, f, b, c, d, a}, Vx = 4, p = f, α= min{Ve}= min{3}, Vx = 4 $\geq$ 3 = α, se elimina a "f" y a todos sus descendientes

L = {b, c, d, a},x = b, Vx = 3, p = a, α= min{ϕ}, NP , Vp = max{$-\infty$, 3} = 3, eliminar nodo

L = {c, d, a}, x = c, Vp = $+\infty$

L = {g, h, c, d, a}, x = g, Vp = $-\infty$

L = {−1, −3, g, h, c, d, a}, Vx = −1, p = g, α= min{ϕ}, NP , Vp = max{$-\infty$, −1} = −1, eliminar nodo

L = {−3, g, h, c, d, a}, Vx = −3, p = g, α= min{ϕ}, NP , Vp = max{−1, −3} = −1, eliminar nodo

L = {g, h, c, d, a}, x = g, Vx = −1, p = c, α = max {Vb} = 3, Vx = −1 $\leq$ 3 = α, se elimina a "c" y a todos sus descendientes

L = {d, a}, x = d, Vp = $+\infty$

L = {i, j, d, a}, x = i, Vp = $-\infty$

L = {4, −2, 1, i, j, d, a}, Vx = 4, p = i, α= min{ϕ}, NP , Vp = max{$-\infty$, 4} = 4, eliminar nodo

L = {−2, 1, i, j, d, a}, Vx = −2, p = i, α= min{ϕ}, NP , Vp = max{4, −2} = 4, eliminar nodo

L = {1, i, j, d, a}, Vx = 1, p = i, α= min{ϕ}, NP , Vp = max{4, 1} = 4, eliminar nodo

L = {i, j, d, a}, x = i, Vx = 4, p = d, α = max{Vb}= max{3} = 3, Vx = 4 $\leq$ 3 = α, NP ,Vp = min{4,4} = 4

L = {j, d, a}, x = j, Vp = $-\infty$entonces eliminamos a "j" y todos sus descendientes

Sistemas de juego

Damas: Chinook terminó el reinado de 40 años (1955 a 1958 y de 1975 a 1991) del campeón mundial Marion Tinsley en 1994.

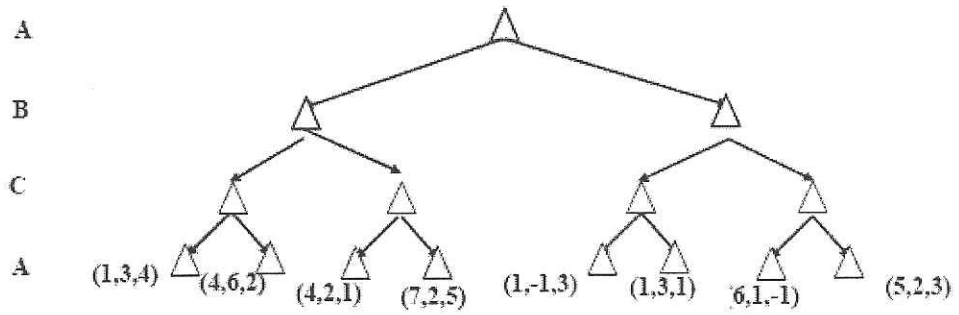
Ajedrez: Deep Blue (IBM) le ganó al campeón mundial Gary Kasparov en una partida de seis juegos en 1997..

Otelo: campeones humanos rechazan competir contra sistemas de cómputo, que son muy buenos.

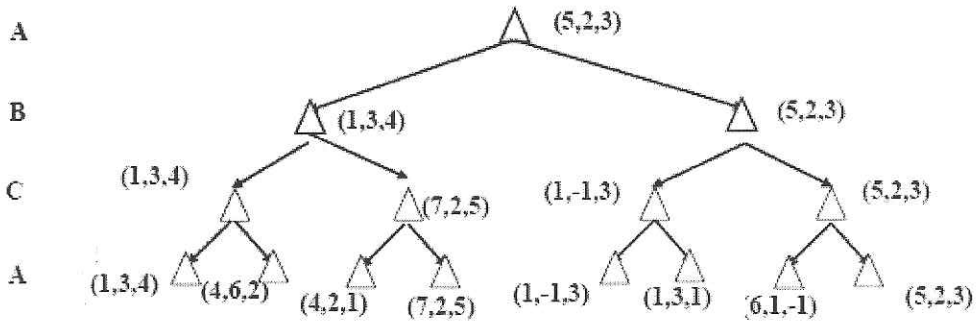
GO: los campeones humanos rechazan competir contra sistemas de cómputo, que son demasiado malos, ya que en GO, b> 300.

4.4. Juegos de múltiples adversarios

En juegos de múltiples adversarios, los valores de MINIMAX se vuelven vectores. Por ejemplo, en la siguiente figura, se representa un juego con tres adversarios: A, B, C. En los nodos terminales se indican los valores para cada jugador (A, B, C).



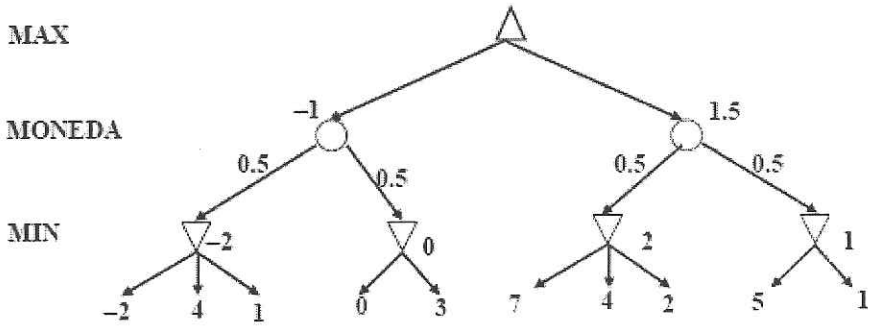
Después de aplicar el algoritmo MINIMAX, los valores de cada nodo se muestran en la figura siguiente:



4.5. Juegos no determinísticos

La introducción de elementos aleatorios, como un dado, en juegos dan lugar a juegos no determinísticos, como el backgamon. La solución de estos juegos mediante métodos de búsqueda requiere la introducción de los posibles valores de ese elemento aleatorio.

Ejemplo, considerando una moneda como elemento aleatorio:



5. Representación del Conocimiento

Las personas conocemos algo, lo tenemos en nuestra mente de alguna forma, y utilizamos ese conocimiento para entender algo nuevo o descifrar algo incomprensible, mediante la inferencia. Por ejemplo, si oímos la siguiente frase:

*Compré una pizza. **La** puse en la mesa de la cocina y media hora después me **la** comí toda.*

Hay dos pronombres "**la**" que requieren una referencia. El primero solamente tiene un antecedente: compré una pizza, por lo que está asociado a pizza. Para el segundo pronombre, hay varios elementos femeninos: pizza, mesa, cocina, hora, y uno solo es comible, la pizza.

5.1. Agentes basados en conocimiento (pensar racionalmente)

Los agentes reflejo encuentran de chiripa una trayectoria de inicio a meta. Toman decisiones basadas en percepciones. Su funcionamiento es bueno sólo si el entorno es totalmente observable.

Los agentes lógicos (basados en conocimiento) combinan su conocimiento general con percepciones actuales para poder inferir aspectos ocultos del estado actual antes de seleccionar sus acciones. De esta forma, en un entorno parcialmente observable pueden almacenar información de las partes del mundo que no les es posible observar.

El conocimiento lo podemos dividir en conocimiento general del dominio y conocimiento de sentido común. Éste último nos permite saber que un objeto en este mundo no puede estar en dos lugares distintos a la vez. El conocimiento general del dominio del juego de ajedrez nos permite saber los movimientos legales de la pieza "rey".

Un agente lógico posee:

- Base de Conocimiento – contenido específico para el dominio
- Máquina de inferencia – algoritmos independientes del dominio

Base de conocimiento

Conjunto de hechos del mundo, oraciones, en un lenguaje formal (lenguaje de representación del conocimiento). La representación es un conjunto de convenciones sobre la forma de describir estos hechos del mundo. La descripción debe utilizar esas convenciones para describir un hecho particular, haciendo explícitos los objetos y sus relaciones importantes y ocultando lo demás.

Los elementos de un lenguaje de representación son: léxico (determina los símbolos permitidos). sintaxis (describe las reglas sobre la forma en que los símbolos se relacionan), semántica (establece una forma de interpretar las descripciones). El lenguaje natural tiene una gran fuerza de expresión lo que permitiría expresar cualquier cosa, sin embargo, es difícil de automatizar por su ambigüedad frecuente. Aún cuando no exista ambigüedad, en su procesamiento automático requiere una vasta cantidad de antecedentes y un amplio conocimiento del mundo. En cambio los lenguajes declarativos no permiten ambigüedad y su semántica define que conclusiones se deducen.

Máquina de Inferencia

La máquina de inferencia aplica las reglas y hechos de la base de conocimiento de acuerdo a algún método de inferencia para llegar a nuevas conclusiones. Estas conclusiones pueden responder a preguntas concretas. Como generalmente la base de conocimiento es muy grande, se requiere un mecanismo de búsqueda en la BC que permita la deducción de forma organizada y eficiente. Otra función importante es mantener la consistencia

5.2. Construcción del agente lógico

Para la BC primero es necesario incorporar el conocimiento, con todo lo que necesita saber.

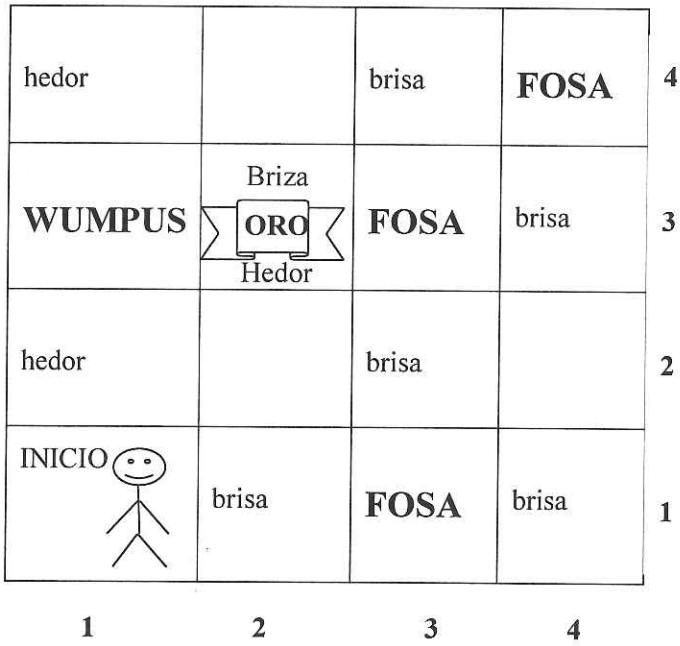
Teniendo todos estos hechos del mundo, el agente lógico puede preguntar a su BC la respuesta, dicha respuesta se deriva mediante inferencia del conocimiento incorporado.

El conocimiento incorporado, independientemente de su implementación, representa el nivel de conocimiento del agente y hasta este punto se considera categórico, es decir, cada hecho es verdadero o falso. En el nivel de implementación se consideran las estructuras de datos y los algoritmos que los manipulan. Un agente lógico que juega ajedrez debe saber las reglas del juego y éstas pueden estar en listas, matrices, etc. La inferencia puede realizarla mediante la manipulación de símbolos, en árboles, etc.

Entonces, el agente lógico debe ser capaz de:

- Representar estados, acciones, etc.
- Incorporar nuevas percepciones (dice a la BC lo que ha percibido)
- Actualizar las representaciones internas del mundo
- Deducir propiedades no percibidas del mundo
- Deducir acciones apropiadas (pregunta a la BC qué acción ejecutar)

Un ejemplo sencillo para mostrar los mecanismos de representación de conocimiento es el Mundo de WUMPUS [Norvig & Russell]



ENTORNO

El entorno es una matriz de 4 x 4, con el inicio en 1,1

Observable: NO
sólo percepciones locales

Determinístico: SI
salidas especificadas exactamente

Episódico: NO
secuencial al nivel de acciones

Estático: SI
Wumpus y las fosas no se mueven

El propósito del agente es tomar el oro. Si entra a una casilla con fosa, el agente no podrá salir de ella. Si entra a la casilla del Wumpus, éste se lo comerá. En las casillas aledañas a la fosa se percibe una brisa y en las casillas aledañas al Wumpus se percibe un hedor. Si el oro está en un casilla hay resplandor en ella. Descripción de percepciones: [BRISA, HEDOR, RESPLANDOR]

Actuadores. El agente puede moverse adelante, atrás, izquierda a 90°, derecha a 90°, los movimientos permiten un solo paso entre casillas, y puede tomar un objeto.

Sensores. Pueden percibir hedor, brisa, resplandor. Si no hay fosa ni Wumpus en la casilla se considera libre (OK). El inicio [1,1] es una casilla libre.

La BC del agente contiene en el inicio las reglas del ambiente. Por ejemplo:

Si hay una fosa en [1,2] o en [2,1] habrá brisa en [1,1]

Posición del agente (casilla)	Percepción	Respuesta de la BC
[1,1]	[nada, nada, nada]	MOVER a [1,2] o [2,1]
[1,2]	[brisa, nada, nada]	FOSA en [2,2] o [1,3] → regresa a [1,1]
[1,1]	[nada, nada, nada]	MOVER a [2,1]
[2,1]	[nada, hedor, nada]	WUMPUS en [3,1] o [2,2] → WUMPUS en [3,1] → MOVER a [2,2]
[2,2]	[nada, nada, nada]	→ MOVER a [3,2]
[3,2]	[brisa, hedor, resplandor]	→ TOMAR oro

EJERCICIO: dar una cuarta columna con la actualización de la BC

Cuando requerimos que cualquier sistema con conocimiento haga algo que no ha sido explícitamente dicho como hacerlo, debe razonar. Por ejemplo, WUMPUS no está en `[1,2]` porque no se detectó hedor en `[1,1]`

5.3. Un lenguaje formal: lógica

Un lenguaje formal además del léxico cuenta con:

- Sintaxis – determina qué expresiones son legales

Por ejemplo, en lógica proposicional.

oración atómica → VERDADERO | FALSO | símbolo proposicional

símbolo proposicional → P | Q | R

- Semántica – qué significan las expresiones legales

Por ejemplo, en la lógica define el valor de verdad de cada oración respecto a cada mundo posible (modelo)

- Sistema de prueba – una forma de manipular expresiones sintácticas para obtener otras expresiones sintácticas, que expresen algo nuevo

Las pruebas podrían ser: obtener conclusiones sobre el mundo (mediante múltiples percepciones), obtener propiedades del siguiente estado (mediante el estado actual y el operador).

La lógica provee una forma de manipular una colección grande de conjuntos mediante la manipulación de descripciones cortas. Cuando se tienen demasiados estados, es deseable una forma de tratar con un conjunto de estados. Por ejemplo: todos los perros, $\forall x$ perro(x)

Implicaciones

La BC implica una oración α si y solo si α es verdadera en todos los mundos donde la BC es verdadera.

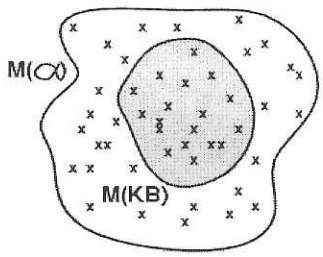
$KB \models \alpha$

La BC que contiene "hay una brisa en [1,2]" y "hay una brisa en [2,3]", la BC implica "hay una fosa en [2,2] o hay una fosa en [1,3]"

La implicación es una relación entre oraciones (sintaxis) que se basa en la semántica.

Modelos

Los modelos son mundos estructurados formalmente con respecto a los cuales se puede evaluar la validez.

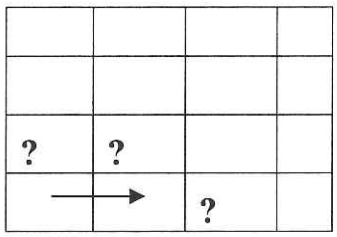


m es un modelo de una oración α si α es verdadera en *m*

$M(\alpha)$ es el conjunto de todos los modelos de α

Entonces, KB $\models \alpha$ si y solo si $M(KB) \subseteq M(\alpha)$

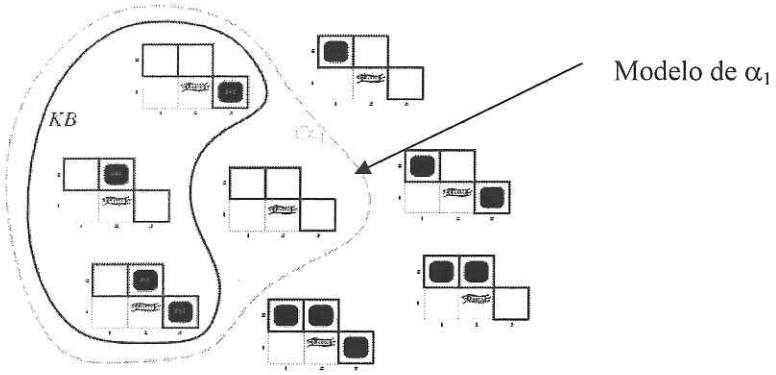
Después de no detectar algo en [1,1] se mueve el agente a [1,2] donde detecta brisa. Los posibles modelos suponiendo solamente fosas son 8 posibles modelos considerando 3 selecciones booleanas.



La BC es falsa en los modelos que contradicen lo que el agente sabe. Por ejemplo, es falsa en todo lo que asegure una fosa en [1,2] porque no detectó brisa en [1,1] ni fosa en [2,1]

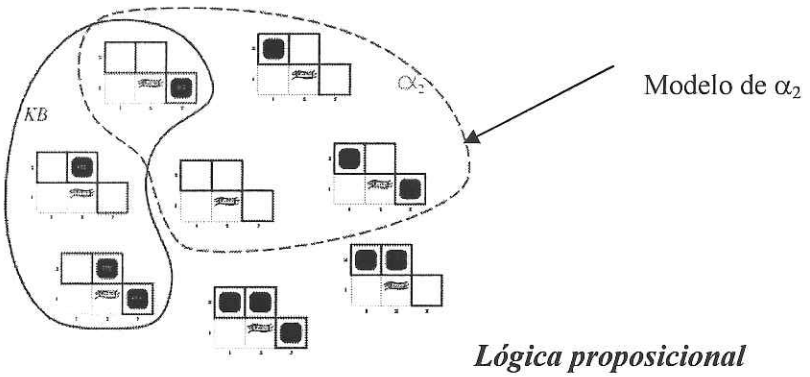
La BC contiene las reglas del mundo de Wumpus más las percepciones. En cada modelo en el que la BC es verdadera, α_1 también

α_1 = "[2,1] está libre", BC $\models \alpha_1$ (no hay fosa en [2,1]), se puede probar con una inferencia conocida como comprobación de modelos, que se muestra a continuación.



En algunos modelos en los que la BC es verdadera, α_2 no lo es, donde

α_2 = "[2,2] está libre" (no hay una fosa en [2,2]), BC $\not\models \alpha_2$



Lógica proposicional

Sintaxis de la lógica proposicional

oración → oración atómica | oración compleja

oración atómica → VERDADERO | FALSO | símbolo proposicional

símbolo proposicional → P | Q | R

oración compleja → ¬ oración | oración ∧ oración | oración ∨ oración | oración ⇒ oración | oración ⇔ oración

Semántica de la lógica proposicional

Consideremos el modelo m_1 = {$FOSA_{1,2}$ = falso, $FOSA_{2,2}$ = falso, $FOSA_{1,3}$ = verdadero}

Donde $FOSA_{i,j}$ es verdadero si hay un hoyo en [i,j]

$BRISA_{i,j}$ es verdadero si hay una corriente de aire en [i,j]

Con 3 símbolos proposicionales hay 2^3 modelos posibles.

La semántica en lógica proposicional debe especificar cómo obtener el valor de verdad de cualquier oración, dado un modelo. Para las oraciones atómicas: *verdadero* es verdadero en todos los modelos, el valor de verdad de cada símbolo proposicional se debe especificar directamente para cada modelo. Ejemplo: $FOSA_{1,2}$ = falso. Para las oraciones complejas: una tabla de verdad. Ejemplo: ¬ $FOSA_{1,2}$ ∧ ($FOSA_{2,2}$ ∨ $FOSA_{1,3}$) evaluada según m_1 es verdadera.

BC

Regla 1) $\neg$FOSA1,1

Regla 2) $\neg$WUMPUS1,1

Regla 3) BRISA1,1 $\Leftrightarrow$ (FOSA1,2 $\vee$ FOSA2,1)

Regla 4) BRISA1,2 $\Leftrightarrow$ (FOSA1,1 $\vee$ FOSA 2,2 $\vee$ FOSA1,3)

Percepciones:

Regla 5) $\neg$BRISA1,1

Regla 6) BRISA1,2

Lógica de primer orden

Sintaxis de la Lógica de primer orden

Constantes:	Juan, 2, NUS,...
Predicados:	Hermano, >,...
Funciones:	RaizCuadrada, piernaIzquierdaDe,...
Variables:	x, y, a, b,...
Conectores:	$\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
Igualdad:	=
Cuantificadores:	$\forall$, $\exists$

Reglas

Oración	→ oración atómica | oración conector oración | cuantificador variables, … oración | $\neg$ oración
oración atómica	→ predicado(término, …) | término = término
término	→ función(término, …) | constante | variable
conector	→ $\wedge$ | $\vee$ | $\Rightarrow$ | $\Leftrightarrow$
cuantificador	→ $\forall$ | $\exists$

Combinaciones posibles de cuantificación universal y existencial

$\forall x \forall y$ padre(x,y) $\equiv \forall y \forall x$ padre(x,y)
$\exists x \exists y$ padre(x,y) $\equiv \exists y \exists x$ padre(x,y)
$\forall x \exists y$ padre(x,y) $\neq \exists y \forall x$ padre(x,y)
$\exists x \forall y$ padre(x,y) $\neq \forall y \exists x$ padre(x,y)

$\Rightarrow$ es el principal conector con $\forall$
$\wedge$ es el principal conector con $\exists$
$\forall x$ padre(x) $\Rightarrow$persona(x)
$\exists x$ padre(x) $\wedge$ persona(x)

Semántica de la lógica de primer orden

En la lógica de primer orden al igual que en lógica preposicional la validez de un postulado se determina dentro de algún contexto. Los símbolos y las constantes se interpretan como objetos, los símbolos de función se interpretan como funciones de objetos a objetos, los símbolos de relación se interpretan como relaciones entre objetos

Una oración es válida si su valor es verdadero en todas las interpretaciones. Por ejemplo: $P \vee \neg P$

Una oración es satisfacible si su valor es verdadero en al menos una interpretación. Por ejemplo: $\neg P$

Una oración es no satisfacible si su valor es falso en todas las interpretaciones. Por ejemplo: $P \wedge \neg P$

Comparación

La lógica proposicional y la de primer orden son declarativas. Ambas permiten información parcial, disyuntiva y negada, a diferencia de la mayoría de las estructuras y bases de datos. Una característica importante es que su significado es composicional, es decir, el significado completo de la oración se deriva de los significados de sus componentes, a diferencia del lenguaje natural. También a diferencia del lenguaje natural, su significado es independiente del contexto.

La lógica proposicional tiene un poder de expresión muy limitado, a diferencia de la lógica de primer orden y en mayor grado del lenguaje natural. Por ejemplo: en todas las casillas adyacentes a una fosa habrá brisa.

La lógica de primer orden no solamente considera hechos como la proposicional sino que permite describir objetos, relaciones, funciones

5.4. Métodos de prueba

Los métodos de prueba se dividen, de forma muy general, en dos tipos:

- Aplicación de reglas de inferencia
  Generación legítima (sólida) de nuevas oraciones a partir de viejas oraciones.
  Prueba: una secuencia de aplicaciones de reglas de inferencia.
  Se pueden utilizar reglas de inferencia como operadores en los métodos de búsqueda.
  Requiere oraciones transformadas a una forma normal.
- Revisión de modelos
  Es la enumeración de una tabla de verdad, donde la cantidad de filas puede ser exponencial en el número de variables.
  El objetivo de la inferencia lógica es decidir si $BC \models \alpha$, por ejemplo si se deduce $FOSA_{2,2}$
  Se puede hacer obteniendo todos los modelos posibles y en 3 de estos modelos la BC es verdadera. Es una enumeración recursiva de un espacio finito de asignaciones a variables.
  Es sólido (porque implementa de forma directa la definición de implicación) y completo (trabaja para cualquier BC y sentencia α y siempre finaliza porque es un conjunto finito).

5.5. Inferencia

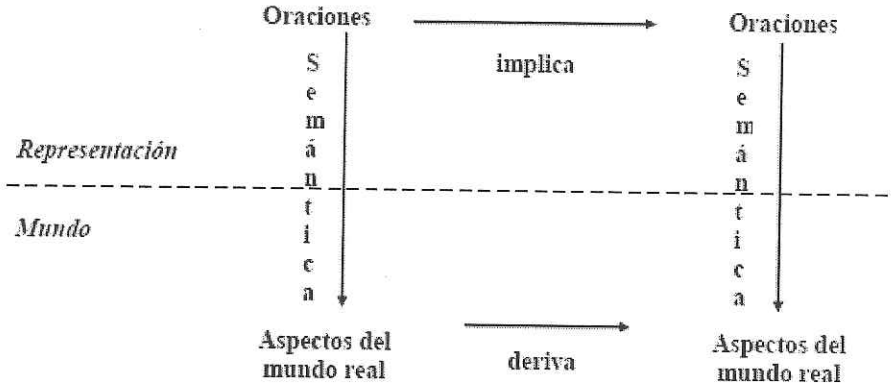
- $BC \vdash_i \alpha$ = la oración α puede derivarse de *BC* mediante el procedimiento *i*
- Sólido: *i* es sólido si siempre que la $BC \vdash_i \alpha$, es verdad que la $BC \models \alpha$
  Si la BC es verdadera en el mundo real, cualquier oración derivada de la BC mediante un procedimiento sólido de inferencia también es verdadera en el mundo real.
- Completo: i es completo si siempre que la $BC \models \alpha$, es verdad que la $BC \vdash_i \alpha$
  El algoritmo puede derivar cualquier oración que se implique.

Para la lógica de primer orden que es más expresiva que la proposicional y con la cual se pueden expresar cosas de interés, existen procedimientos de inferencia sólidos y completos. Estos

procedimientos de inferencia contestarán cualquier pregunta cuya respuesta se derive de lo contenido en la BC.

α es verdadera.

Un proceso de inferencia opera con la sintaxis pero se corresponde con la relación del mundo real, según la cual algún aspecto del mundo real es cierto en virtud de que otros aspectos del mundo real lo son.



Reglas de Inferencia

Modus Ponens para oraciones en forma Horn[6] es completo

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Resolución

$$\frac{\alpha \vee A \qquad \neg A \vee \beta}{\alpha \vee \beta}$$

Eliminación Universal

$$\frac{\forall v \ \alpha}{\text{SUST}(\{v/t\}, \alpha)}$$

donde $\{v/t\}$ es la fórmula para remplazar cada ocurrencia de v en α por t

Eliminación Existencial

$$\frac{\exists v \ \alpha}{\text{SUST}(\{v/k\}, \alpha)}$$

donde k es un símbolo constante nuevo, que no aparece en la BC

∧–Eliminación

$$\frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha \wedge \beta}{\beta}$$

∧–Introducción

[6] La forma Horn corresponde a una cláusula (disyunción de literales) con a lo sumo un literal positivo

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta}$$

Modus Ponens generalizado

Para oraciones atómicas p_1', p_2' ... y q

$$\frac{p_1', p_2', \ldots, p_n', \qquad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\text{SUBST}(\{\theta/q\})}$$

Donde hay una sustitución θ tal que: $\text{SUBST}(\{\theta/ p_i'\}) = \text{SUBST}(\{\theta/ p_i\})$

Ejemplo p_1' = MásRico(Juan, Pedro)

p_2' = MásRico(Pedro, María)

$p_1 \wedge p_2 \Rightarrow q$ = MásRico(x,y) $\wedge$ MásRico(y,z) $\Rightarrow$ MásRico(x,z)

SUBST({θ/q}) = SUBST({x/Juan, y/Pedro, z/María})

Se concluye que MásRico(Juan, María)

Resolución generalizada

$$\frac{p_1 \vee p_2 \vee \ldots \vee p_n \qquad m_1 \vee m_2 \vee \ldots \vee m_k)}{(p_1 \vee \ldots \vee p_{i-1} \vee p_{i+1} \vee \ldots \vee p_n \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_k)\,\theta}$$

Donde UNIFICAR(p_i, $\neg m_j$) = θ

EJEMPLO:

Los empresarios que salen en Forbes son más ricos que los políticos.
Hay un diputado que es más rico que todos los obreros
Carlos es un empresario que sale en Forbes
Juan y Pedro son obreros

Probar que Carlos es más rico que Juan

1) $\forall x \forall y$ EmpresarioForbes(x) $\wedge$ Político(y) $\Rightarrow$ MásRico(x,y)
2) $\exists x$ Diputado(x) $\wedge$ ($\forall z$ Obrero(z) $\Rightarrow$ MásRico(x,z))
3) EmpresarioForbes (Carlos)
4) Obrero(Juan)
5) Obrero(Pedro)

Hacen falta las siguientes reglas de conocimiento general

6) $\forall x$ Diputado(x) $\Rightarrow$ Político(x)
7) $\forall x \forall y \forall z$ MásRico(x,y) $\wedge$ MásRico(y,z) $\Rightarrow$ MásRico(x,z)

Eliminación de existencial en 2:

8) Diputado(UnoMás) $\wedge$ ($\forall z$ Obrero(z) $\Rightarrow$ MásRico(UnoMás, z))

∧–Eliminación en 8

9) Diputado(UnoMás)

10) ∀z Obrero(z) ⇒ MásRico(UnoMás, z)

Eliminación Universal en 10:

11) Obrero(Juan) ⇒ MásRico(UnoMás, Juan)

12) Obrero(Pedro) ⇒ MásRico(UnoMás, Pedro)

Modus Ponens en 4 y 11:

13) MásRico(UnoMás, Juan)

Eliminación Universal en 5:

14) Diputado(UnoMás) ⇒ Político(UnoMás)

Modus Ponens en 9 y 14:

15) Político(UnoMás)

Eliminación Universal en 1:

16) EmpresarioForbes(Carlos) ∧ Político(UnoMás) ⇒ MásRico(Carlos, UnoMás)

∧–Introducción con 3 y 15

17) EmpresarioForbes(Carlos) ∧ Político(UnoMás)

Modus Ponens en 16 y 17:

18) MásRico(Carlos, UnoMás)

Eliminación Universal en 7:

19) MásRico(Carlos, UnoMás) ∧ MásRico(UnoMás, Juan) ⇒ MásRico(Carlos, Juan)

20) MásRico(Carlos, UnoMás) ∧ MásRico(UnoMás, Pedro) ⇒ MásRico(Carlos, Pedro)

∧–Introducción con 13 y 18

21) MásRico(Carlos, UnoMás) ∧ MásRico(UnoMás, Juan)

Modus Ponens en 21 y 19:

22) MásRico(Carlos, Juan)

Unificación

Para aplicar el método de Resolución es necesario definir el proceso de Unificación. La *Unificación* trata de encontrar las sustituciones generales de variables en dos términos, de tal manera que después de aplicar las sustituciones encontradas a ambos términos, los términos sean iguales.

Para conseguir que cada argumento de un literal sea coincidente con su homólogo en el otro literal, debemos buscar una *sustitución* que nos permita emparejarlos. La única condición que debe reunir esta sustitución es que ha de aplicarse a todo el literal, es decir, que la sustitución afecta a todo el literal, y no sólo al argumento en cuestión. Por decirlo de una manera sencilla, las sustituciones se van *arrastrando* a lo largo del proceso de unificación.

Ejemplo:

Para unificar P(x, x) con P(y, z):
Primera sustitución: (y/x)

Resultado: P(y, y) P(y, z)
Segunda sustitución: (z/y)

Resultado: P(z, z) P (z, z)

La sustitución resultante es la composición de las sustituciones: s = {z/y , y/x}

Eskolemizar

El método de transformación a forma de *Skolem* elimina los cuantificadores existenciales.

La fórmula siguiente:

$$\forall x_1 \; \forall x_2 \; \ldots \; \forall x_n \; \exists y \; k(x_1, x_2, ..., x_n, y)$$

es satisfacible si y sólo si

$$\forall x_1 \; \forall x_2 \; \ldots \; \forall x_n \; k(x_1, x_2, ..., x_n, g(x_1, x_2, ..., x_n))$$

Donde *g* es una nueva función constante n-aria, llamada función Skolem, que no aparece en *k*

Por ejemplo:

$$\exists t \; \forall x \; \forall y \; \exists z \; \forall u \; \exists w \; F(t, x, y, z, u, w)$$

es satisfacible si y sólo si

$$\forall x \; \forall y \; \forall u \; F(c, x, y, g(x, y), u, h(x, y, u))$$

Donde *c* es una constante Skolem que no aparece en F, *g* y *h* son funciones Skolem que no aparecen en F.

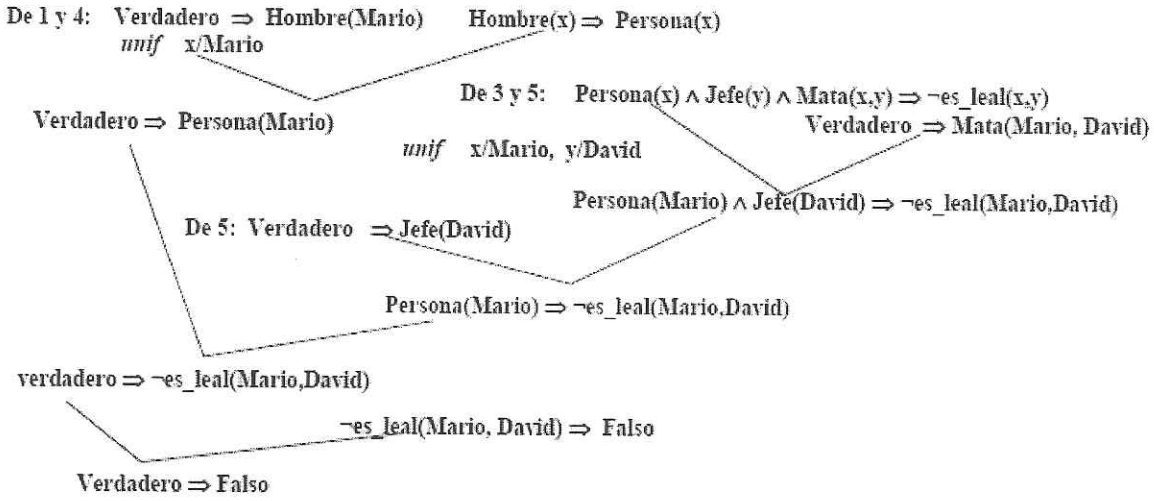
Conversión a CNF

Para aplicar el método de Resolución es necesario conocer el proceso de conversión a forma clausal, ya que las cláusulas con las que se trabaja en esta técnica deben tener una forma específica.

Para transformar una fórmula a *Forma Normal* aplicaremos las siguientes reglas:

- Eliminar todos los símbolos de equivalencia ($\Leftrightarrow$), sustituyéndolos por una implicación a la derecha y una implicación a la izquierda: $\alpha \Leftrightarrow \beta$ por $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$
- Eliminar todas las implicaciones ($\Rightarrow$), remplazando $\alpha \Rightarrow \beta$ por $\neg\alpha \vee \beta$
- Reducir el ámbito de las negaciones ($\neg$), a un único término, usando las siguientes propiedades:
  $\neg (\neg p)=p$
  Leyes de Morgan: $\neg (a \vee b)= \neg a \wedge \neg b$ y $\neg (a \wedge b)= \neg a \vee \neg b$
  $\neg \forall x \; P(x) = \exists x \; \neg P(x)$
  $\neg \exists x \; P(x) = \forall x \; \neg P(x)$
- Normalizar las variables de los cuantificadores, de forma que cada uno esté ligado con una única variable. Para ello se pueden renombrar las variables.
- Desplazar cuantificadores a la izquierda
- Eliminar cuantificador universal, Eskolemizando
- Eliminar cuantificadores universales
- Aplicar distributivamente ($\wedge$ sobre $\vee$) y simplificar

Ejemplo:



$BRISA_{1,1} \Leftrightarrow (FOSA_{1,2} \vee FOSA_{2,1})$

1) $(BRISA_{1,1} \Rightarrow (FOSA_{1,2} \vee FOSA_{2,1})) \wedge ((FOSA_{1,2} \vee FOSA_{2,1}) \Rightarrow BRISA_{1,1})$
2) $(\neg BRISA1,1 \vee FOSA1,2 \vee FOSA2,1) \wedge (\neg (FOSA1,2 \vee FOSA2,1) \vee BRISA1,1)$
3) $(\neg BRISA1,1 \vee FOSA1,2 \vee FOSA2,1) \wedge ((\neg FOSA1,2 \vee \neg FOSA2,1) \vee BRISA1,1)$
4) $(\neg BRISA1,1 \vee FOSA1,2 \vee FOSA2,1) \wedge (\neg FOSA1,2 \vee BRISA1,1) \wedge (\neg FOSA2,1 \vee BRISA1,1)$

Resolución

Un procedimiento para responder a una pregunta es eficaz si siguiendo ese procedimiento se obtiene la respuesta correcta a la pregunta en un número finito de pasos. Un procedimiento de inferencia completo usando la resolución es la refutación (prueba por contradicción o reducción al absurdo). La idea es que para probar P se asume ¬P y se prueba una contradicción.

$$(BC \wedge \neg P \Rightarrow Falso) \Leftrightarrow (BC \Rightarrow P)$$

Ejemplo.

Mario es un hombre
Mario es chiapaneco
Todos los chiapanecos son mexicanos
David es el jefe
Todos los mexicanos son leales al jefe o lo odian
Cada uno es leal a alguien
Las personas sólo matan jefes a quienes no son leales
Mario mató a David

Primero añadir a la BC: ¬es_leal(Mario, David) ⇒ *Falso* para responder ¬es_leal(Mario, David)
1) Hombre(Mario)
2) Jefe(David)
3) Persona(x) ∧ Jefe(y) ∧ Mata(x,y) ⇒ ¬es_leal(x,y)
4) Hombre(x) ⇒ Persona(x)

5) Mata(Mario, David)

5.6. Encadenamiento

Encadenamiento hacia adelante

El encadenamiento hacia adelante es sólido y completo para BC en forma de Horn

El encadenamiento hacia adelante deriva cada oración atómica que es implicada por la BC:

- Alcanza un punto fijo donde no se pueden derivar nuevas oraciones atómicas
- Considera el estado final como un modelo m, asignando falso o verdadero a los símbolos
- Cada cláusula en la BC original es verdadera en m

$a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge \ldots \wedge a_n \Rightarrow b$

- Entonces m es un modelo de la BC
- Si BC $\models q$, q es verdadera en cada modelo de la BC, incluyendo m

Las pruebas comienzan con los axiomas o premisas de la BC, deriva nuevas oraciones con MPG hasta que la meta se deriva. Esto es un procedimiento de inferencia con encadenamiento hacia adelante porque se mueve hacia adelante, de la BC a la meta.

Ejemplo: Queremos saber si Pluto come huesos

BC A todos los perros les gustan los huesos, los perros comen lo que les gusta, Pluto es un perro.

BC en LPO: $\forall x$ perro(x) $\Rightarrow$ gustar(x, Hueso)

2. $\forall x \forall y$ (perro(x) $\wedge$ gusta(x,y)) $\Rightarrow$ come(x,y)
3. perro(Pluto)

Inferencia

Usando MPG con (1) y (3) se deriva: 4. gusta(Pluto, Hueso)

Usando MPG con (3), (4) y (2) se deriva come(Pluto, Hueso)

Encadenamiento hacia atrás

La idea es hacer la inferencia hacia atrás, partiendo de la pregunta q para probar q mediante la BC. El proceso se lleva a cabo de la siguiente forma:

- Revisar si q ya está en la BC
- Probar mediante la BC todas las premisas de alguna regla que concluya q
- Evitar círculos, revisar si una nueva submeta está ya en la pila de la meta
- Evitar el trabajo repetido, revisar si una submeta ya ha sido probada verdadera o si ya ha fallado

El encadenamiento hacia atrás es completo para BC en forma de Horn. La prueba comienza con la pregunta meta, encuentra implicaciones que permitan probarla y enseguida prueba cada uno d elos antecedentes de la implicación y continúa trabajando hacia atrás hasta que se llega a los axiomas que son verdaderos.

Ejemplo: Queremos saber si Pluto come huesos

Inferencia.

La meta empata con la parte derecha de la cláusula (2) entonces hay que probar las nuevas submetas perro(Pluto) y gusta(Pluto, Hueso)

perro(Pluto) empata con (3), una submeta solucionada

gusta(Pluto, Hueso) empata con (1) si x= Pluto

Como se solucionaron todas las submetas, se termina

Comparación

El encadenamiento hacia adelante está dirigido por los datos, es un proceso inconsciente, automático. Puede hacer un montón de trabajo que no es relevante para la meta. Por ejemplo: reconocimiento de objetos, decisiones de rutina.

El encadenamiento hacia atrás está dirigido por la meta. Es apropiado para la solución de problemas, respuesta a preguntas: ¿dónde está el libro? ¿cómo ingresar a la licenciatura?

La complejidad del encadenamiento hacia atrás puede ser mucho menor que lineal en tamaño.

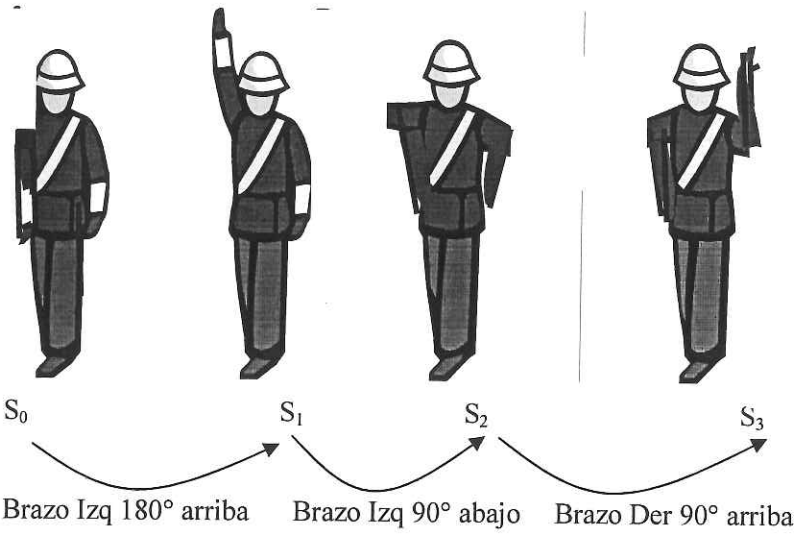
5.7. Representación de cambios en el mundo

Para mantener un registro de los cambios en el mundo, un método que se ha utilizado es el Cálculo de Situaciones (McCarthy and Hayes, 1969). Este Cálculo de Situaciones es un aspecto de la aproximación lógica a la Inteligencia Artificial. El razonamiento sobre resultados de las acciones es primordial para un agente basado en conocimiento. Para ello, el agente requiere mantener un registro de posiciones y localidades mediante la lógica. El Cálculo de Situaciones también permite una cierta representación del tiempo mediante esas situaciones. Se extiende la lógica de primer orden al añadir un argumento de situación a cada predicado no eterno.

Se considera que una situación es una instantánea del mundo en algún momento. A partir de hechos de situaciones y reglas generales de los efectos de las acciones es posible inferir algo sobre futuras situaciones. Las situaciones están conectadas por los resultados de las acciones. Los hechos se mantienen en las situaciones.

Resultado(Brazo-arriba,S_0)	El resultado es estar en una nueva situación S_1 como efecto de subir un brazo estando en la situación S_0
Resultado(Tomar,S_5)	El resultado es una nueva situación S_6 como efecto de tomar algo estando en la situación S_5

En el Cálculo de Situaciones, las acciones y las situaciones son términos lógicos. Las situaciones constan de: una situación inicial S_0, y todas las situaciones resultantes de la acción en S_0. Los flujos son funciones y predicados que varían de una situación a la siguiente, por ejemplo: ¬Está(Agente, S_0) como efecto de moverse hacia adelante. También hay predicados que no cambian, eternos, como Oro(O1)



Para describir el cambio en el Cálculo de Situaciones se requieren axiomas.

1) Axioma de posibilidad

Describe la posibilidad de realizar una acción.

Posición(BrazoIzq,0°,s) ∧ MovimBrazo(0°, 180° arriba) ⇒ Posible(MovBrazIzq(0°, 180° arriba),s)

2) Axioma de efecto

Describe los cambios debidos a una acción

Posible(MovBrazIzq(0°, 180° arriba),s) ⇒ Posición(BrazIzq,180°arriba, Resultado(MovBrazIzq(0°, 180° arriba),s))

3) Axioma de Marco

Pero no todo es el resultado de una acción, hay cosas que permanecen igual. El agente puede mover sus brazos pero su uniforme sigue intacto o al mover el brazo izquierdo, el brazo derecho sigue en la misma posición. El problema del "marco" (frame problem) es cómo representar todo lo que permanece igual. Una posibilidad es el Axioma de Marco que describe lo que no cambia debido a las acciones.

Posición(b,x,s) ∧ (b ≠ BrazoIzq) ∧ ¬Sostener(o,s) ⇒ Posición(b,x, Resultado(MovBrazIzq(y, z),s))

Con estos axiomas, el agente puede deducir el resultado de una secuencia de acciones, o planear (encontrar una secuencia de acciones) para obtener un efecto específico.

En el mundo real, se presentan algunos problemas:

El problema de posibilidad: las descripciones de las acciones requieren advertencias sin fin. Por ejemplo: si el agente sufre un estirón en el brazo no podrá moverlo, si se le atora el brazo en la ropa, si no hay espacio libre para el movimiento, si ..., etc.

El problema de ramificación: las acciones reales tienen muchas consecuencias secundarias. Cuando el agente mueve el brazo, también mueve. su ropa, el reloj de pulsera, sus guantes, el polvo, etc.

El problema del marco: puede hacer que la inferencia sea exponencial por la cantidad de cosas que no cambian. Si hay F flujos y A acciones, se necesitan AF axiomas de marco para describir los

objetos que no cambian. Una solución es describir cómo cada flujo cambia en el tiempo, mediante el axioma de estado sucesor:

P es verdadero $\Leftrightarrow$ [una acción hace que P sea verdadero $\vee$ P ya es verdadero y ninguna acción lo hace falso]

$Pos(a,s) \Rightarrow (At(Agent,y,Result(a,s)) \Leftrightarrow (a = Go(x,y)) \vee (At(Agent,y,s) \wedge a \neq Go(y,z))$

$Posible(a,s) \Rightarrow (Posición(BrazIzq, x, Resultado(a,s)) \Leftrightarrow (a = MovBrazIzq (x,y)) \vee (Posición (BrazIzq,y,s) \wedge a \neq MovBrazIzq (y,z))$

Con este axioma, el siguiente estado está especificado completamente por el estado actual, y cada efecto de la acción se indica una sola vez.

Ejercicio

Para el agente en el mundo de Wumpus, con la BC inicial:

En (Agente, [1,1], So)

En (Oro, [1,2], So)

Responder si existe una S tal que Sostener(Oro,S)

6. Razonamiento con incertidumbre

Hasta aquí, hemos visto ejemplos simples de razonamiento o de alguna forma hemos delineado la inferencia. Por ejemplo si sabemos que las ballenas desdentadas son cetáceos y que los cetáceos tienen pulmones, entonces se puede responder a la pregunta ¿tienen pulmones las ballenas? Con muy poco razonamiento se puede responder dicha pregunta.

En la lógica propositiva y de primer orden, la validez de una conclusión se mantiene al llegar nueva información, cuando se tiene toda la información necesaria para la inferencia que se quiere hacer. Pero esto puede cambiar si algunas precondiciones son suposiciones hipotéticas que se invalidan con nueva información y entonces tendrá que retractarse porque la conclusión ya no es válida. Por ejemplo, si en algún momento se descubre una especie de cetáceo que no cuente con pulmones, tal vez por evolución, se retractaría "los cetáceos tienen pulmones".

Si esto sucede, y se llega a que alguna conclusión es inválida, se tendría que analizar qué otras conclusiones son inválidas. Lo mismo con las acciones, si alguna acción no se puede realizar, igualmente se tendría que analizar qué otras acciones tampoco pueden realizarse. Esto implica una búsqueda exhaustiva en la BC, que puede llegar a representar un enorme esfuerzo en tiempo y memoria.

6.1. Razonamiento incierto

Desafortunadamente el mundo es un lugar incierto. Cualquier sistema de I A que intente modelar y razonar en este mundo debe ser capaz de tratar con:

- un estado incompleto (falta de conocimiento). Por ejemplo: cuando no se conoce exactamente cómo funciona un órgano del cuerpo humano se está razonando con un estado incompleto.
- inconsistencia (ambigüedades y contradicciones). Por ejemplo: un juez puede estar razonando con ambigüedades y contradicciones de testigos.
- cambios (actualización de conocimiento en el tiempo). Por ejemplo: el primer planeta fuera del sistema solar fue detectado hasta 1995 y de ahí ha seguido confirmándose la existencia de otros planetas.

Además debemos considerar que

- el lenguaje de representación a menudo es impreciso. Por ejemplo: cómo podemos representar que la mayoría de las aves vuelan, no todas.
- la información es aproximada. Por ejemplo: una ave vuela si está viva, si no tiene alas rotas, si no es pingüino, si no es avestruz, si no está llena de chapopote, si ...
- no existen relaciones absolutas de causa-efecto. Por ejemplo: investigaciones científicas afirman que las personas que usan marihuana tienen una mayor ascendencia en desórdenes psiquiátricos comparados con los que no la usan.

Algunos de los métodos que pueden enfrentarse con esta incertidumbre son: lógica por omisión, herencia, métodos estadísticos.

6.2. Razonamiento no monotónico

A diferencia del razonamiento monotónico donde no se retrae ninguna aseveración y por lo tanto va aumentando la BC con la inferencia, el razonamiento humano no sigue esta estructura monotónica ya que muchas veces brincamos a conclusiones que nos permitan sobrevivir. Por ejemplo, debemos hacer suposiciones sobre cosas que no conocemos con certeza, vemos una persona que se desmaya pero no sabemos exactamente qué le pasa, sin embargo actuamos para reanimarla de alguna forma. No podemos anticipar todos los posibles resultados de un plan, nos preparamos para llegar a una

determinada hora a la Facultad y resulta que se detuvo el metro, que hubo una manifestación, que hubo un choque, que ....

EJEMPLO

Una persona pone una demanda porque asevera que María, Patricia o Raquel le robaron una pulsera de diamantes. La tenía el lunes y la dejó en su escritorio, el martes que regresó temprano ya no estaba y solamente María, Patricia y Raquel pueden acceder a esa oficina.

Las declaraciones de los involucrados nos llevan a la siguiente BC:

María no robó la pulsera

Patricia no robó la pulsera

Raquel no robó la pulsera

María o Patricia o Raquel robó la pulsera

Las tres primeras oraciones se basan en las siguientes declaraciones:

- María dice que estuvo ocupada en la oficina de Recursos Humanos y que el Sr. Martínez puede confirmarlo.
- Patricia dice que estuvo ayudándole a su cuñada en su oficina.
- Raquel dice que salió de la oficina y fue sola al partido de futbol de su equipo favorito.

La cuarta oración se basa en que no forzaron la entrada y solamente ellas tres tienen llave; además el edificio se cierra después de cierta hora por lo que se tiene el horario en que sucedió el robo y solamente María, Patricia o Raquel pudieron robar la pulsera. Y ésta es la declaración con mayor grado de creencia.

De esta forma, la BC no es consistente por lo que es necesario revisar nuestros grados de creencia en cada sospechosa, revisando a quién creerle menos. María y Patricia tienen testigos, Raquel es la única que no cuenta con una justificación comprobable. Así que si cambiamos 3) por Raquel robó la pulsera, la BC es consistente.

Pasados unos días aparece en la televisión un video del partido de futbol que muestra varias escenas donde aparece Raquel. Con lo cual debemos retractarnos de 3) Raquel robó la pulsera. De esta forma la BC decrece. Así que es necesario volver a revisar nuestros grados de creencia en cada oración.

Razonamiento por omisión

Es una forma común del razonamiento no monotónico donde se obtienen conclusiones basadas en lo que es más posible que sea verdad.

La regla que introduce la Lógica por omisión es:

$$\frac{A \quad B}{C}$$

Que establece que si A es deducible y es consistente suponer B entonces se puede concluir C. Frecuentemente toma la siguiente forma, con justificación y consecuencia iguales:

$$\frac{A \quad B}{B}$$

$$\frac{\text{Ave(x)} \quad \text{puede\_volar(x)}}{\text{puede\_volar(x)}}$$

Normalmente, los pájaros vuelan.

Típicamente, los pájaros vuelan.

Si x es un pájaro, entonces supóngase por omisión que x vuela.

6.3. Herencia

La herencia es una propiedad importante de las redes semánticas. Cuando un objeto pertenece a una clase hereda todas las propiedades de esa clase.

Redes Semánticas

Las redes semánticas son una alternativa a la lógica de predicados como una forma de representación del conocimiento. Esta representación gráfica permite almacenar el conocimiento. Los nodos representan objetos del dominio y los arcos representan las relaciones entre esos objetos. Por ejemplo:

Representa el siguiente conocimiento.

es (Toño, Tigre)

es (Juan, Humano)

propietario (Juan, Titi)

es (Titi, Loro)

color (Loro, Verde)

es (Humano, Mamífero)

es (Loro, Ave)

tiene (Mamíferos, Pelo)

tiene (Aves, Plumas)

es (Animal vertebrado, Mamífero)

es (Animal vertebrado, Ave)

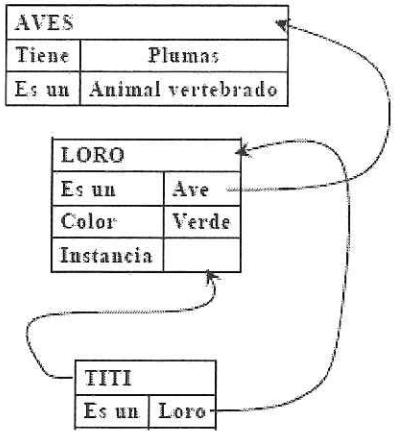
Esta representación tiene imprecisiones, no hay distinción entre nodos representando clases y nodos representando objetos individuales. Estas imprecisiones no suceden en lógica de primer orden. Por ejemplo, para el hecho "Juan es propietario de Titi", su representación en lógica de primer orden es: $\exists x$ (loro(x) $\wedge$ propietario (Juan, x)), en la red semántica se debería especificar un ave individual que posee Juan.

En este ejemplo, Titi hereda la propiedad de tener plumas. Además de Titi, otros loros también serán verdes. De esta forma se puede realizar inferencia.

Marcos

Otra forma de representación corresponde a los marcos, donde a diferencia de las redes semánticas, en lugar de arcos hay ranuras que representan atributos del objeto. El número de ranuras depende de cada clase, del número de propiedades.

La herencia se desarrolla mediante las ranuras a nivel de clase y a nivel de instancia. Las ranuras de clase representan atributos que son comunes a todos los miembros de la clase. Cuando la ranura tiene un valor a nivel de instancia el valor del atributo varía entre miembros de esa clase. Las ranuras pueden contener valores o apuntadores.



6.4. Métodos estadísticos

La teoría de probabilidad nos permite hacer decisiones racionales. Por ejemplo: para saber cuál es el mejor transporte para ir a trabajar nos ayudaría conocer algunos datos como: qué medio de transporte resulta más rápido en la ciudad a las 9 de la mañana, cuál es la probabilidad de un percance en ese medio, etc.

Probabilidades

Un evento tiene una probabilidad. Si consideramos una secuencia de experimentos y vemos el número de veces que un evento particular ocurre y comparamos contra el número total de experimentos, encontramos una proporción. Esta proporción es una forma de calcular la probabilidad de ese evento.

En un juego de cartas, donde se van tomando una a una, P(as de trébol) = 1/52

Axioma: $P(A) + P(\neg A) = 1$

Probabilidad conjunta

Si A y B son dos eventos, su probabilidad conjunta se denota: P(A,B). Por ejemplo: P(as) es la probabilidad de que la carta que se toma sea un as, P(trébol) es la probabilidad de que la carta que se toma sea del palo de trébol. P(as, trébol) es la probabilidad de que la carta que se toma sea un as de trébol.

$$P(A, B) = P(A \mid B)\, P(B)$$

Donde P(A, B) es la probabilidad conjunta, que también se denota $P(A \cap B)$

Por ejemplo:

A es la P(as de trébol) = 1/52, pero si sabemos B es la P(as), entonces P(A|B) = 1/4

P(A|B) = P(A,B) / P(B) = (1/52) / (4/52)

Probabilidades condicionales

La probabilidad de un evento puede cambiar después de conocer otro evento:

Probabilidad del evento *A* dado el evento *B* es

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Probabilidad del evento B dado el evento A es

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

Reacomodando y combinando las dos ecuaciones, tenemos

$$P(A \mid B)\, P(B) = P(A \cap B) = P(B \mid A)\, P(A)$$

Dividiendo ambos lados por P(B), dado que no es cero, obtenemos la regla de Bayes:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

$$P(A|C) + P(\neg A|C) = 1$$

Más sobre dependencias:

Si A = r y B= w,s

P(r|w,s) = P(r,w,s) | P(w,s)

Pero si A = w y B= r,s

P(w,r,s) = P(w|r,s) P(r,s) = P(w|r,s) P(r|s) P(s)

Entonces:

P(r|w,s) = P(r,w,s) / P(w,s) = P(w|r,s) P(r|s) P(s) / (P(w|s) P(s))

= P(w|r,s) P(r|s) / P(w|s)

También

Si A = w,r y B= s

P(w,r,s) = P(w,r|s) P(s)

Y por regla de encadenamiento P(w,r,s) = P(w|r,s) P(r|s) P(s)

Entonces:

$P(w,r|s)\ P(s) = P(w|r,s)\ P(r|s)\ P(s)$

$P(w,r|s) = P(w|r,s)\ P(r|s)$

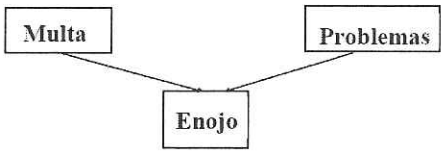
Independencias

Un evento es equivalente a una variable con un valor específico, es decir, la instanciación de una variable es un evento.

Si un evento no tiene nada que ver con otro evento, son independientes. Si A es independiente de B, $P(A|B) = P(A)$. Por ejemplo: tirar un dado es independiente de tirar otro dado.

Eventos dependientes pueden volverse independientes dados ciertos eventos. Dos eventos A y B son condicionalmente independientes dado C si `P(A| B,C) = P(A| C)`.
Los eventos A y B son independientes si y solo si: $P(A, B) = P(A)\ P(B)$, que es equivalente a $P(A|B) = P(A)$ y $P(B|A) = P(B)$, cuando $P(A, B) > 0$
Por ejemplo: A es el evento de tirar un dado, B es el evento de tirar otro dado, entonces $P(B|A) = P(B)$

En el siguiente ejemplo, Multa y Problemas son dos eventos independientes pero cuando se da Enojo, se vuelven dependientes porque compiten por explicar Enojo.



Distribución de Probabilidad Conjunta

Son asignaciones probabilísticas a todas las proposiciones del dominio. Por ejemplo, si en un micromundo solamente tenemos dos variables booleanas: Multa y Enojo, su distribución de probabilidad es la siguiente:

	MULTA	¬ MULTA
ENOJO	0.2	0.06
¬ ENOJO	0.1	0.65

Con 4 eventos atómicos, es decir, con 4 posibles asignaciones de valores particulares a todas las variables, que corresponden a la especificación total del estado del dominio. Con estos valores se puede calcular cualquier aseveración probabilística acerca del dominio.

$P(\text{Enojo}) = 0.26$

$P(\text{Multa}) = 0.3$

$P(\text{Multa} \vee \text{Enojo}) = 0.36$

$P(\text{Enojo} \mid \text{Multa}) = P(\text{Enojo} \wedge \text{Multa}) / P(\text{Multa}) = 0.66$

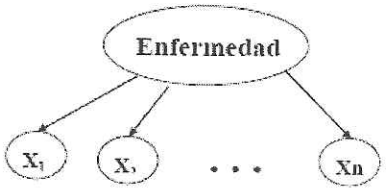
Para el dominio con tres variables booleanas: Enojo, Multa y Problemas, se tienen 2^3 valores.

Redes bayesianas

También se conocen como Redes de Creencias, Redes Causales. Las redes bayesianas se pueden utilizar para construir modelos empleando el conocimiento causal.

Las redes bayesianas permiten disminuir la cantidad de valores de eventos atómicos al considerarse suposiciones de independencia condicional.

Por ejemplo, una Red Bayesiana para diagnóstico médico tiene un solo nodo que representa si el paciente tiene una enfermedad particular, y las otras variables X_1, X_2..., X_n son hijas directas del nodo de enfermedad.



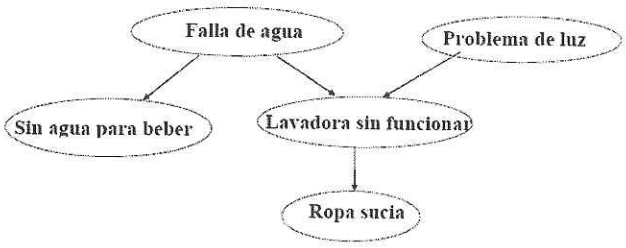
La distribución conjunta sobre todas las variables está dada por:

$$P(D,X_1, \ldots ,X_n) = P(D)\prod^{n}_{i=1} P(X_i \mid D)$$

En el diagnóstico médico, si E es un conjunto de síntomas, por ejemplo: tos, estornudos, dolor de cabeza, etc. y H es una enfermedad, por ejemplo: resfriado común, N1H1, SARS, gripa, el problema de diagnóstico es encontrar un H (enfermedad) tal que P (H|E) sea máximo.

La red bayesiana es una gráfica dirigida acíclica que representa las relaciones probabilísticas de dependencia e independencia entre las variables aleatorias del dominio. Los nodos representan variables aleatorias, cada nodo corresponde a una variable. Las aristas entre dos nodos representan la influencia probabilística, si X y Y son dos nodos conectados con una arista dirigida de X a Y, X tiene una influencia directa en Y. Cada nodo tiene una tabla de probabilidad condicional (TPC) en términos de los nodos padre.

Ejemplo



Las variables aleatorias son cada una un conjunto exhaustivo de posibilidades mutuamente exclusivas. Por ejemplo, la edad de una persona, la presión sanguínea de una persona. La instanciación de una variable es un evento.

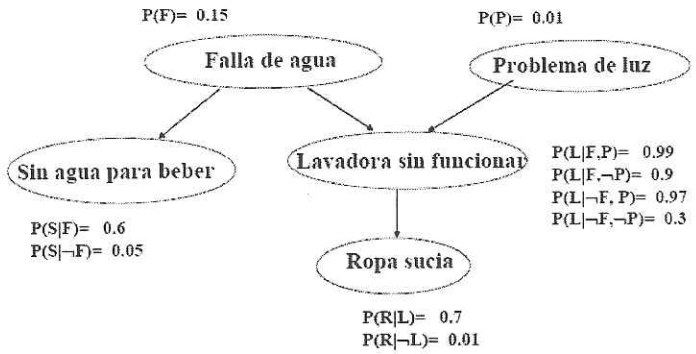
Un conjunto de variables son independientes si y sólo si en todas las posibles instanciaciones de las variables se mantiene la independencia.

Probabilidades en las redes Bayesianas - los nodos están cuantificados con distribuciones de probabilidades. Los nodos sin padres tienen probabilidades a priori. Los nodos con padres tienen tablas de probabilidades condicionales (TPC) del nodo dado sus padres.

Dada una red bayesiana y sus TPC se puede calcular probabilidades como $P(H \mid E_1, E_2, \ldots \ldots , E_n)$, donde $H, E_1, E_2, \ldots , E_n$ son variables aleatorias del dominio.

Ejemplo:

La probabilidad de que la Lavadora no funcione dado que hay Problema de luz y falla de agua: P(L| P, F)



De dos modos podemos entender la semántica de las redes bayesianas, como la representación de la distribución de probabilidad conjunta y como la codificación de un conjunto de postulados de independencia condicional.

Las entradas de la evaluación de la red bayesiana son un conjunto de evidencias, por ejemplo: E = {Falla-de-agua = verdadero}. Las salidas de la evaluación de las redes bayesianas son del tipo P(Xi=verdadero| E), donde Xi es una variable de la red. Por ejemplo, P(Sin-agua-para-beber= verdadero| Falla-de-agua) es la probabilidad de que no haya agua para beber dado que hay una falla de agua.

Para que la inferencia probabilística sea práctica es necesario hacer suposiciones de independencia. Las redes bayesianas explícitamente modelan las relaciones de independencia de las variables del dominio. Para ejemplificar cómo las suposiciones de independencia disminuyen la distribución de probabilidad conjunta, partimos de la regla de encadenamiento. La regla de encadenamiento permite expresar una distribución de probabilidad conjunta como un producto de probabilidades condicionales:

$P(X_1, X_2, ..., X_n)$

$=P(X_1)\, P(X_2, X_1, ..., X_n \mid X_1)$

$=P(X_1)\ P(X_2|X_1)\ P(X_3, X_4, ..., X_n| X_1, X_2)$

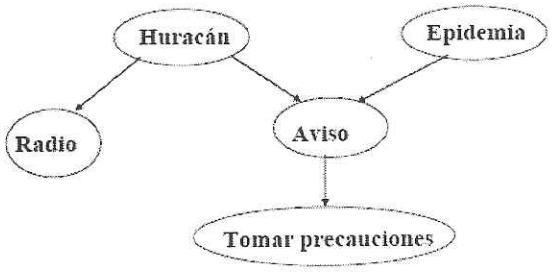
$=P(X_1)\ P(X_2|X_1)\ P(X_3|X_1,X_2)\ P(X_4, ..., X_n|X_1,X_2,X_3)... ...$

$= P(X_1)\ P(X_2|X_1)\ P(X_3|X_1, X_2)\ \ ...\ \ P(X_n| X_1, ...X_{n-1})$

Ejemplo:

Supongamos que se da un *aviso* en la comunidad cuando hay un *huracán* o una epidemia. Si se da ese aviso las familias toman precauciones para sobrevivir. Además cuando hay un huracán desde la oficina central se envían anuncios por *radio*.

Hay cinco variables aleatorias: *aviso* (A), *huracán* (H), *epidemia* (E), *tomar precauciones* (TP), *radio* (R). A continuación se representa este dominio con la red Bayesiana siguiente:



Para esta red:

P(E) — la probabilidad a priori de Epidemia

P(H) — la probabilidad a priori de Huracán

P(A | H,E) — la probabilidad de que se dé el aviso bajo cualquier circunstancia

P(TP | A) — la probabilidad de que se tomen precauciones para cada valor de Aviso

P(R | H) — la probabilidad de que se anuncie por el radio cualquier Huracán

Como son 5 variables habría 2^5 valores en la tabla de probabilidades conjuntas, en la red bayesiana hay 10 valores únicamente

Cálculo de la Probabilidad Conjunta

Dada una red Bayesiana, donde X_1, X_2..., X_n son los nodos etiquetados de tal forma que sólo los nodos con índice menor a *i* pueden tener trayectorias dirigidas a X_i, se puede calcular:

$P(X_i \mid X_1 \ldots X_{i-1}) = P(X_i \mid \text{padres}(X_i))$

Esta probabilidad está disponible en la red Bayesiana. Por lo tanto, $P(X_1, X_2 \ldots, X_n)$ puede calcularse a partir de las probabilidades disponibles en la red Bayesiana.

Las redes Bayesianas dan una forma compacta de representar la probabilidad conjunta en un conjunto de variables arbitrarias.

Inferencia en redes bayesianas

La principal tarea de inferencia es calcular la probabilidad condicional de un conjunto de variables dados los valores precisos de algunas variables evidencia. Por ejemplo: P(dolor|golpe) Cada nodo puede servir como variable de evidencia o de evento.

Un agente que emplea redes bayesianas toma de sus percepciones los valores de algunas variables aleatorias, y enseguida pregunta a la red los valores de otras variables.

Entre las inferencias que se pueden realizar están:

- Inferencia Causal (descendente, de efectos a causas).
  Ejemplo: P(Ropa_sucia| Lavadora_sin_funcionar)
- Inferencia de diagnóstico (ascendente, de efectos a causas).
  Ejemplo: P(Lavadora_sin_funcionar| Ropa_sucia)

Inferencia causal o descendente

Por ejemplo: P(Aviso| Epidemia)

- Expandir las probabilidades condicionales del nodo consulta (Aviso) en términos de la probabilidad conjunta del nodo consulta y todos sus padres que no son evidencia, dada la evidencia

P(Aviso|Epidemia) = P(Aviso, Huracán| Epidemia) + P(Aviso, ¬Huracán| Epidemia)

- Expresarla como la probabilidad del nodo consulta condicionada por todos sus padres

Probabilidades conjuntas: $P(A|B) = P(A,B) / P(B)$

$P(\text{Aviso, Huracán}|\ \text{Epidemia}) = P(\text{Aviso, Huracán, Epidemia}) / P(\text{Epidemia}) = P(\text{Aviso}|\ \text{Huracán, Epidemia})\ P(\text{Huracán}|\ \text{Epidemia})\ P(\text{Epidemia})$

Entonces $P(\text{Aviso, Huracán, Epidemia}) = P(\text{Aviso}|\ \text{Huracán, Epidemia})\ P(\text{Huracán}|\ \text{Epidemia})$

Pero de la red bayesiana $P(\text{Huracán}|\ \text{Epidemia}) = P(\text{Huracán})$

$P(\text{Aviso}|\ \text{Epidemia}) = P(\text{Aviso, Huracán}|\ \text{Epidemia}) + P(\text{Aviso, }\neg\text{Huracán}|\ \text{Epidemia})$

$P(\text{Aviso}|\ \text{Epid}) = P(\text{Aviso}|\ \text{Huracán, Epid})\ P(\text{Huracán}) + P(\text{Aviso}|\ \neg\text{Huracán, Epid})\ P(\neg\text{Huracán})$

Inferencia de diagnóstico o ascendente

Por ejemplo: $P(\neg\text{Epidemia}|\ \neg\text{Aviso})$

Regla de Bayes: $P(\neg\text{Epidemia}|\ \neg\text{Aviso}) = P(\neg\text{Aviso}\ |\neg\text{Epidemia})\ P(\neg\text{Epidemia}) /P(\neg\text{Aviso})$ donde $P(\neg\text{Aviso}\ |\neg\text{Epidemia})$ es una inferencia causal

Para el valor faltante $P(\neg\text{Aviso}\ |\neg\text{Epidemia})$:

$P(\neg\text{Aviso}|\ \neg\text{Epid}) = P(\neg\text{Aviso}|\ \text{Huracán, }\neg\text{Epid})\ P(\text{Huracán}) + P(\neg\text{Aviso}|\ \neg\text{Huracán, }\neg\text{Epid})\ P(\neg\text{Huracán})$

Para obtener $P((\neg\text{Aviso})$:

$1 = P(\neg\text{Epidemia}|\ \neg\text{Aviso}) + P(\text{Epidemia}\ |\neg\text{Aviso})$
$= P(\neg\text{Aviso}|\ \neg\text{Epid})\ P(\neg\text{Epid}) / P(\neg\text{Aviso}) + P(\neg\text{Aviso}|\ \text{Epid})\ P(\text{Epid}) / P(\neg\text{Aviso})$

$P(\neg\text{Aviso}) = P(\neg\text{Aviso}|\ \neg\text{Epid})\ P(\neg\text{Epid}) + P(\neg\text{Aviso}|\ \text{Epid})\ P(\text{Epid})$

Donde: $P(\neg\text{Aviso}|\ \neg\text{Epid})$ y $P(\neg\text{Aviso}|\ \text{Epid})$ son causales

7. Aprendizaje

Según el diccionario, aprendizaje es adquirir el conocimiento de algo por medio del estudio o de la experiencia; Concebir algo por meras apariencias o con poco fundamento; Tomar algo en la memoria.

Hasta este punto, solamente se han considerado problemas donde el estado completo del mundo se conoce de antemano. Sin embargo, es muy raro que se pueda especificar completamente el estado del mundo a priori. Por lo que frecuentemente el agente debe ser capaz de aprender sobre el mundo mediante observaciones.

Se conoce como aprendizaje automático el área de la Inteligencia artificial que desarrolla técnicas que permitan a las computadoras aprender. Muchas técnicas en aprendizaje automático se derivan de los esfuerzos de psicólogos de hacer más precisas sus teorías de aprendizaje en animales y humanos mediante modelos computacionales. Parece posible también que los conceptos y las técnicas exploradas por los investigadores en aprendizaje automático puedan aclarar ciertos aspectos del aprendizaje biológico.

La habilidad para aprender puede estar basada en experiencia. El agente, basándose en su BC y su experiencia será capaz de producir nuevo conocimiento. A partir de una secuencia de percepciones y su BC, aprenderá hechos que son consistentes con ambos o que no solamente se derivan de ellos.

7.1. Aprendizaje inductivo

Mucho del aprendizaje que actualmente se desarrolla es de naturaleza inductiva. Dados ciertos datos experimentales, el agente aprende los principios generales que rigen esos datos y a partir de ellos puede predecir correctamente datos futuros. Por ejemplo, después de decirle a un niño que ciertos animales son perros, el niño puede aprender el concepto de "perro" y reconocer perros que previamente no había visto como tales.

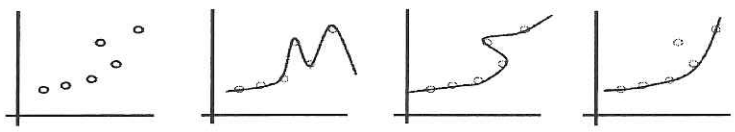
Cuando la variable que tratamos de predecir es continua el problema de aprendizaje se llama problema de regresión, en cambio cuando tiene sólo un pequeño número de valores distintos se llama problema de clasificación.

Aprendizaje puramente inductivo

El aprendizaje se puede ver como aprendizaje de la representación de una función.

Dada una colección de pares (entrada, salida) llamados ejemplos $\{(x_1; f(x_1)); \ldots., (x_n; f(x_n))\}$, para una función f, produce una hipótesis, una función h (representación compacta) que aproxime f

Generalmente hay bastantes hipótesis diferentes consistentes con los ejemplos. En la figura siguiente se muestran tres posibles funciones para los 6 ejemplos de la colección.



Predisposición (Bias) en el aprendizaje

Cualquier clase de preferencia por una hipótesis h sobre otra se llama predisposición

La predisposición es ineludible, sólo la selección del formalismo para describir h introduce ya una predisposición. La predisposición es necesaria, puede haber miles de hipótesis y se debe hacer una selección.

7.2. Aprendizaje supervisado

En el aprendizaje supervisado, el algoritmo trabajará con ejemplos de entrenamiento etiquetados, es decir, los ejemplos tienen las etiquetas de la propiedad que se intenta predecir. Por ejemplo, si se intenta clasificar gatos, los datos de animales tienen una etiqueta que indica si son gatos o no.

Aprendizaje de árboles de decisión

La forma más simple de aprendizaje inductivo supervisado corresponde al aprendizaje de árboles de decisión. Es un método más apropiado para entradas discretas. Un árbol de decisión es un operador booleano que toma como entrada un conjunto de propiedades que describen un objeto o una situación, y la salida es una decisión: sí/no o una clasificación.

Está representado por un árbol en el cual:

- cada nodo interno corresponde a una prueba del valor de una de las propiedades
- cada nodo hoja especifica el valor que se regresará (sí o no, o clase) si se llega a esa hoja

Es trivial construir un árbol de decisión que esté de acuerdo con un conjunto de entrenamiento dado. Sin embargo, este tipo de árbol memorizará simplemente los ejemplos dados.

Queremos un buen árbol de decisión, un árbol que extrapole un patrón común a partir de los ejemplos. Queremos que el árbol de decisión clasifique correctamente todos los ejemplos posibles, no solamente los del conjunto de entrenamiento sino ejemplos nuevos.

Búsqueda de árboles de decisión

Generalmente hay varios árboles de decisión que describen el mismo predicado meta. ¿Cuál debemos preferir? La navaja de Ockham: prefiera siempre la descripción más simple, es decir, el árbol más pequeño.

Problema: búsqueda a través del espacio de árboles posibles para encontrar el más pequeño, es posible pero tarda tiempo exponencial.

Solución: aplicar alguna heurística simple que lleve a árboles pequeños (aunque no sea el más pequeño)

Idea principal: comenzar construyendo el árbol probando en su raíz un atributo que separe mejor el conjunto de entrenamiento en clases homogéneas.

Construcción del Árbol: procedimiento general

1. Elegir para el nodo raíz el atributo que mejor divida el conjunto de ejemplos de entrenamiento en conjuntos homogéneos

2. Si el atributo elegido tiene n valores posibles, dividirá E en n conjuntos. Añadir una rama i al nodo raíz para cada conjunto E_i

3. Para cada rama i:

    3.1. Si E_i contiene sólo ejemplos positivos, añadir una hoja SI a la rama.

    3.2. Si E_i contiene sólo ejemplos negativos, añadir una hoja NO a la rama

    3.3. Si no, añadir un nodo no-hoja a la rama y aplicar el procedimiento recursivamente al nodo con E_i como conjunto de entrenamiento

Para elegir el atributo que mejor divida el conjunto de ejemplos de entrenamiento en conjuntos homogéneos, se requiere un método que indique cuál atributo es mejor. La Teoría de la Información permite obtener medidas para evaluar los atributos. La Teoría de la Información estudia las leyes matemáticas que rigen los sistemas diseñados para comunicar o manipular información. En esta teoría se definen medidas cuantitativas de la información y la capacidad de los sistemas para transmitir, almacenar, y procesar la información.

Una medida adecuada es la cantidad esperada de información de la teoría de información de Shannon y Weaver (1949) que vincula las nociones de desorden e información. A mayor desorden en un conjunto se requiere más información para adivinar correctamente un elemento del conjunto.

El valor de desorden V asociado con la partición de atributos es la suma de probabilidades sobre todas las particiones S_i de los valores de información I de la partición de clase $\{P(S_i), N(S_i)\}$

S es el conjunto de ejemplos
S_i es una partición de S resultante del atributo i
N es el conjunto de ejemplos negativos
P es el conjunto de ejemplos positivos

$P \cup N = S$

$$V\left[\{S_i \mid 1 \le i \le n\}\right] = \sum_{i=1}^{N} \frac{|S_i|}{|S|} I\left(\{P(S_i), N(S_i)\}\right)$$

Donde $$I\left(\{P(S_i), N(S_i)\}\right) = \log_2(|S|) - \frac{|P|}{|S|}\log_2|P| - \frac{|N|}{|S|}\log_2|N|$$

Estas fórmulas corresponden al aprendizaje de un concepto, es decir, de dos clases (positiva y negativa), para más clases se requieren las siguientes fórmulas:

$$I(\{X_i \mid 1 \le i \le n\}) = \log_2|X| - \sum_{i=1}^{n} \frac{|X_i|}{|X|}\log_2|X_i|$$

$$V(\{S_i \mid 1 \le i \le n\}) = \sum_{i=i}^{n} \frac{|S_i|}{|S|} I(\{X_i \mid 1 \le i \le n)\})$$

Donde X_i son las clases.

Ejemplo:

Dada la siguiente tabla que representa ejemplos de clasificación de películas de cine, construir el árbol de decisión resultante de aplicar el método de

ID	Nacionalidad	Género	Actores	Guión	Director	*Clase*
1	USA	Acción	Estrellas	original	bueno	buena
2	USA	Drama	actor de verdad	adaptado	bueno	buena
3	Europa	Drama	actor de verdad	adaptado	bueno	buena
4	USA	aventuras	actor de verdad	original	normal	buena
5	Europa	aventuras	actor de verdad	adaptado	bueno	buena
11	Asia	Acción	Normal	original	normal	mala
12	Asia	Drama	Normal	original	normal	mala
13	Europa	Drama	Normal	original	por encargo	mala

14	Europa	comedia	Estrellas	original	por encargo	mala
15	USA	Drama	Estrellas	original	normal	mala

Como solamente se aprende un concepto (buena +, mala -) X solamente es X1(+) y X2(-)

Partición por Género

S = {{1} {11}, {2,3} {12,13,15}, {4,5} {}, {} {14}}

$\underbrace{+\ -}_{\text{Acción}}$ $\underbrace{+\quad -}_{\text{Drama}}$ $\underbrace{+\ -}_{\text{Aventura}}$ $\underbrace{+\ -}_{\text{Comedia}}$

Partición por Actores

S = {{1} {14,15}, {2,3,4,5} {}, {} {11,12,13}}

$\underbrace{+\quad -}_{\text{Estrellas}}$ $\underbrace{+\quad -}_{\text{Actor verdad.}}$ $\underbrace{+\quad -}_{\text{Normal}}$

Partición por Guión

S = {{1,4} {11,12,13,14,15}, {2,3,5} {}}

$\underbrace{+\qquad -}_{\text{Original}}$ $\underbrace{+\ -}_{\text{Adaptado}}$

Partición por Director

S = {{1,2,3,4,5} {}, {4} {11,12,15}, {} {12,13}}

$\underbrace{+\quad -}_{\text{Bueno}}$ $\underbrace{+\quad -}_{\text{Normal}}$ $\underbrace{+\ -}_{\text{Por encargo}}$

Partición por Nacionalidad

S = {{1,2,4} {15}, {3,5} {13,14}, {} {11,12}}

$\underbrace{+\quad -}_{\text{USA}}$ $\underbrace{+\quad -}_{\text{Europa}}$ $\underbrace{+\ -}_{\text{Asia}}$

Atributo Nacionalidad

$$\frac{|S_{USA}|}{|S|} = \frac{4}{10}\left[\log_2(4) - \frac{3}{4}\log_2(3) - \frac{1}{4}\log_2(1)\right] = 0.324$$

$$\frac{|S_{Europa}|}{|S|} = \frac{4}{10}\left[\log_2(4) - \frac{2}{4}\log_2(2) - \frac{2}{4}\log_2(2)\right] = 0.4$$

$$\frac{|S_{Asia}|}{|S|} = \frac{2}{10}\left[\log_2(2) - (0) - \frac{2}{2}\log_2(2)\right] = 0$$

Desorden (Nacionalidad) = 0.324 + 0.4 + 0 = 0.724

De la misma forma, para los demás atributos:

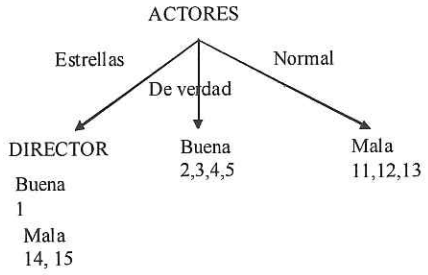
Desorden (Género) = 0.685

Desorden (Actores) = 0.276

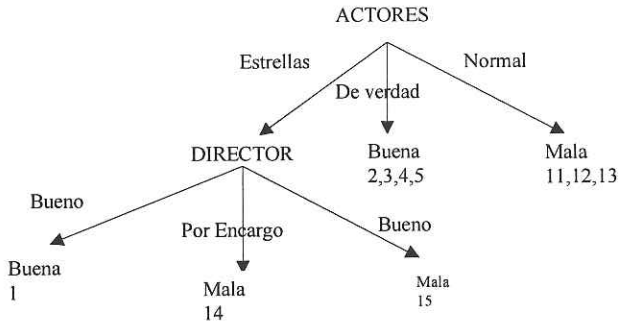
Desorden (Guión) = 0.6041

Desorden (Director) = 0.324

Con estos valores se escoge el primer atributo: Actores



Ahora se tiene un nuevo problema con tres ejemplos y un atributo menos. Los ejemplos son 1,14 y 15 y los atributos son: Género, Guión, Director y Nacionalidad. Se vuelve a aplicar el método y se obtiene el siguiente Árbol de Decisión:



Problemas al construir Árboles de Decisión

- Ruido. Dos ejemplos de entrenamiento pueden tener valores idénticos para todos los atributos, pero estar clasificados de forma diferente
- Sobre representación. Los atributos irrelevantes pueden hacer distinciones falsas entre ejemplos de entrenamiento
- Datos faltantes. El valor de algunos atributos de algunos ejemplos de entrenamiento pueden faltar
- Atributos con múltiples valores. La ganancia de información de un atributo con muchos valores diferentes tiende a ser distinta de cero aun cuando el atributo sea irrelevante.
- Atributos valorados de forma continua. Deben volverse discretos para poderlos usar. De todas las posibles formas de discretizar, unas son mejores que otras en cuanto al propósito de clasificar

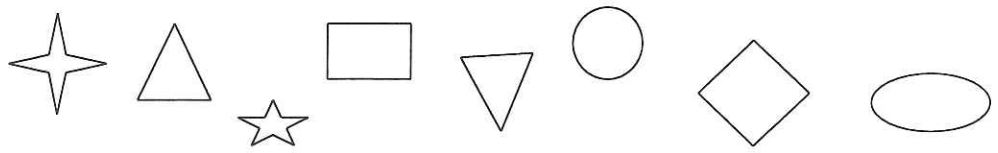
7.3. Aprendizaje no supervisado

En el aprendizaje no supervisado, los ejemplos de entrenamiento no están etiquetados, es decir, no existe una clasificación de los ejemplos. Los algoritmos no supervisados más comunes agruparán los ejemplos mediante la similaridad y/o la disimilaridad entre ejemplos. Este método se conoce como *clustering*.

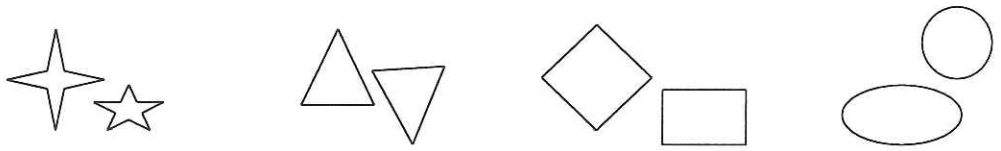
Clustering

Mediante este método grandes conjuntos de ejemplos de entrenamiento se agrupan en conjuntos más pequeños de ejemplos similares.

Por ejemplo, consideremos las figuras siguientes:



Estamos interesados en agrupar las 8 figuras en 4 grupos, considerando la similitud en sus ángulos podríamos clasificarlos de la forma siguiente:



Los métodos de clustering intentan encontrar grupos naturales de ejemplos, basándose en alguna similaridad. Encuentran el centro de cada grupo (centroide) y de esta forma evalúan la distancia de cada ejemplo al centroide de cada grupo. La salida del algoritmo de clustering es una descripción de los centroides de los clusters con el número de ejemplos en cada cluster.

Algoritmo K-means

Inicialmente el número de clusters se conoce o se escoge, será k

1) Seleccionar un conjunto k de ejemplos como centroides
2) Para cada ejemplo, asignarlo al cluster donde sea más parecido al centroide
3) Volver a calcular los centroides
4) Regresar a (1) hasta que no haya modificaciones de centroides o hasta cierto límite

Ejemplo

	X	Y
EJEM 1	1	1.8
EJEM 2	1.1	1.6
EJEM 3	0.8	1.3
EJEM 4	1	0.9
EJEM 5	1.5	0.1
EJEM 6	1.1	0.1

Centroides:

EJEM 4

	Dist. a E4	Dist. a E7
EJEM 1	0.9	**0.781025**
EJEM 2	**0.707106781**	1
EJEM 3	**0.447213595**	1.140175
EJEM 4	**0**	1.581139
EJEM 5	**0.943398113**	2.507987
EJEM 6	**0.806225775**	2.376973

EJEM 7	0.5	2.4
EJEM 8	1.5	0.4

EJEM 7

EJEM 7	1.58113883	**0**
EJEM 8	**0.707106781**	2.236068

Donde **Dist** es la distancia euclidiana en coordenadas cartesianas entre ejemplos

Nuevos clusters : (EJEM 2,EJEM 3, EJEM 4,EJEM 5,EJEM 6,EJEM 8)

(EJEM 1,EJEM 7)

Nuevos Centroides:

a) X=1.125, Y=0.7

b) X=0.83333, Y=1.7

	Dist. a (a)	**Dist. a (b)**
EJEM 1	1.107079491	**0.194365**
EJEM 2	0.900347155	**0.2848**
EJEM 3	0.682367203	**0.401386**
EJEM 4	**0.235849528**	0.817177
EJEM 5	**0.707548585**	1.733333
EJEM 6	**0.600520607**	1.62207
EJEM 7	1.811249569	**0.775314**
EJEM 8	**0.480234318**	1.460974

Nuevos clusters : (EJEM 4,EJEM 5,EJEM 6,EJEM 8)

(EJEM 1,EJEM 2,EJEM 3,EJEM 7)

Nuevos Centroides:

a) X=1. 275,Y=0.375

b) X=0.85, Y=1.775

	Dist. a (a)	**Dist. a (b)**
EJEM 1	1.451292527	**0.152069**
EJEM 2	1.237436867	**0.305164**
EJEM 3	1.039831717	**0.477624**
EJEM 4	**0.59266348**	0.887764
EJEM 5	**0.35531676**	1.796698
EJEM 6	**0.32596012**	1.693554
EJEM 7	2.168236611	**0.716327**
EJEM 8	**0.226384628**	1.520896

Termina el ciclo al no haber cambios de ejemplos en los clusters.

8. BIBLIOGRAFIA

Básica

Russell, Stuart and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall (3rd Edition) 2009

Ginsberg, M. *Essentials of Artificial Intelligence.* Morgan Kaufmann Publishers Inc., 1993.

Nilsson, Nils. *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann, San Mateo, Ca., 1998

Dean, T., James Allen, and Yiannis Aloimonos. *Artificial Intelligence: theory and practice.* The Benjamin/Cummings Publishing Co., 1995.

Rich, E. and Kevin Knight. *Artificial Intelligence, Second Edition*, McGraw Hill Book Company, 1991.

Complementaria

Bielawsky, L.; Lewand, R., *Intelligent Systems Design; Integrating Expert Systems, Hypermedia, and Database Technologies,* John Wiley & Sons, Inc., 1991

Boden, M. A. (Ed.). The Philosophy of Artificial Intelligence. Oxford University Press. 1990

Bowen, K., *ProLog and Expert Systems Programming*, McGraw-Hill, 1991

Garcia, O. N.; Chien, Y.-T., Editores, *Knowledge-Based Systems, Fundamentals and Tools*, IEEE Computer Society Press, 1991

Goertzel, Ben; Pennachin, Cassio, eds. *Artificial General Intelligence*, Springer, 2006

Iyengar, S.; Elfes, A.; Editores, *Autonomous Mobile Robots: Perception, Mapping, and Navigation, Volumes 1 and 2,* IEEE Computer Society Press, 1991

Jackson, P., *Introduction To Expert Systems, Second Edition*, Addison-Wesley Publishing Company, 1990

Jones, T. *Artificial Intelligence: A Systems Approach*, Jones and Bartlett, Sudbury, Mass., 2009.

Luger, G. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th ed., Addison-Wesley, Reading, Mass., 2009.

Mitchell, Tom. Machine Learning. McGraw Hill, 1997

Poole, D. and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, Cambridge University Press, Cambridge, UK, 2010.

Poole, David, Alan Mackworth, and Randy Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, 1998

Schalkoff, R., *Artificial Intelligence*, McGraw-Hill, 1990

Winston, P. H. *Artificial Intelligence*, 3rd ed., Addison-Wesley, Reading, Mass., 1992.